JUNOS™ Internet Software Configuration Guide

Policy Framework

Release 5.6

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA
408-745-2000
www.juniper.net

Part Number: 530-008940-01, Revision 1

JUNOS Internet Software Configuration Guide: Policy Framework, Release 5.6

Writer: Joshua Kim
Editor: Cris Morris
Covers and template design: Edmonds Design

Revision History
27 December 2002—First edition.

The information in this document is current as of the date listed in the revision history.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer or otherwise revise this publication without notice.

Products made or sold by Juniper Networks (including the M5, M10, M20, M40, M40e, and M160 routers, T320 router, T640 routing node, and the JUNOS software) or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,333,650, 6,359,479, and 6,406,312.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. The JUNOS software has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

SOFTWARE LICENSE

The terms and conditions for using this software are described in the software license contained in the acknowledgment to your purchase order or, to the extent applicable, to any reseller agreement or end-user purchase agreement executed between you and Juniper Networks. By using this software, you indicate that you understand and agree to be bound by those terms and conditions.

Generally speaking, the software license restricts the manner in which you are permitted to use the software and may contain prohibitions against certain uses. The software license may state conditions under which the license is automatically terminated. You should consult the license for further details.

For complete product documentation, please see the Juniper Networks Web site at www.juniper.net/techpubs.

Table of Contents

About This Manual

Part 1

Policy Framework

Chapter 1 Policy Framework Overview 3

Part 2
Routing Policies

List of Figures

List of Figures

List of Tables

List of Tables

About This Manual

This chapter provides a high-level overview of the *JUNOS Internet Software Configuration Guide: Policy Framework:*

Objectives

This manual provides an overview of the policy framework for the JUNOS Internet software and describes how to configure the policies on the router.

This manual documents Release 5.6 of the JUNOS Internet software. To obtain additional information about the JUNOS software—either corrections to information in this manual or information that might have been omitted from this manual—refer to the software release notes.

To obtain the most current version of this manual and the most current version of the software release notes, refer to the product documentation page on the Juniper Networks Web site, which is located at http://www.juniper.net/.

To order printed copies of this manual or to order a documentation CD-ROM, which contains this manual, please contact your sales representative.

Audience

This manual is designed for network administrators who are configuring a Juniper Networks router. It assumes that you have a broad understanding of networks in general, the Internet in particular, networking principles, and network configuration. This manual assumes that you are familiar with one or more of the following Internet routing protocols: Border Gateway Protocol (BGP), Routing Information Protocol (RIP), Intermediate System-to-Intermediate System (IS-IS), Open Shortest Path First (OSPF), Internet Control Message Protocol (ICMP) router discovery, Internet Group Management Protocol (IGMP), Distance Vector Multicast Routing Protocol (DVMRP), Protocol-Independent Multicast (PIM), Multiprotocol Label Switching (MPLS), Resource Reservation Protocol (RSVP), and Simple Network Management Protocol (SNMP).

Document Organization

This manual is divided into several parts. Each part describes a major functional area of the JUNOS software, and the individual chapters within a part describe the software commands of that functional area.

This manual contains the following parts and chapters:

- Preface, “About This Manual” (this chapter), provides a brief description of the contents and organization of this manual and describes how to contact customer support.
- Part 1, “Policy Framework,” provides an overview of the JUNOS policy framework, which includes routing policies and firewall filter policies.
  - Chapter 1, “Policy Framework Overview,” discusses the policies that comprise the JUNOS policy framework, the router flows they affect, the fundamental architecture they share, and their similarities and differences.
- Part 2, “Routing Policies,” describes the use of routing policies to control routing information between the routing protocols and routing tables, and between the routing tables and the forwarding table.
  - Chapter 2, “Routing Policy Framework Overview,” provides an overview of the routing policy framework, default routing policies and actions, user-defined routing policies, and routing policy evaluation.
  - Chapter 3, “Routing Policy Configuration Statements,” lists the statements that you can include when you create routing policies.
  - Chapter 4, “Configure Routing Policy,” describes how to configure a routing policy, which includes defining match conditions and actions and applying the policy. This chapter also includes routing policy configuration examples and a network case study.
  - Chapter 5, “Configure Extended Match Conditions,” discusses how to configure autonomous system (AS) path regular expressions, communities, prefix lists, route lists, and subroutines as match conditions for a routing policy and provides examples.

- Chapter 6, “Configure Extended Actions,” describes how to prepend an AS path, set up class of service (CoS), change the default BGP route flap-damping parameters, and set up per-packet load balancing.

- Chapter 7, “Summary of Routing Policy Configuration Statements,” explains the routing policy configuration statements.

- Part 3, “Firewall Filters,” describes the use of firewall filters, including policers, to control packets transiting the router to a network destination and packets destined for or sent by the router.

  - Chapter 8, “Firewall Filter Overview,” provides an overview of firewall filters and describes firewall filter match conditions and actions, the ordering of terms in a firewall filter configuration, and the default actions.

  - Chapter 9, “Firewall Filter Configuration,” explains how to configure a firewall filter, including defining match conditions and actions and applying the filter. It also describes accounting profiles and filter-based forwarding.

  - Chapter 10, “Policer Overview,” provides an overview of a policer and describes rate limits, applying policer statements, and policers stored as templates.

  - Chapter 11, “Policer Configuration,” explains how to configure policers, including specifying rate limiting, policer actions, and applying the policy to filter configurations and directly to interfaces. This chapter also includes policer configuration examples.

  - Chapter 12, “Summary of Firewall Filter and Policer Configuration Statements,” explains the firewall filter and policer configuration statements.

- Part 4, “Traffic Sampling and Forwarding,” describes the use of firewall filters to perform statistical monitoring of traffic

  - Chapter 13, “Traffic Sampling and Forwarding Overview,” provides an overview of sampling and forwarding, including monitoring and saving traffic information, as well as per-flow load balancing, port mirroring, and Domain Name System (DNS) and Trivial File Transfer Protocol (TFTP) forwarding.

  - Chapter 14, “Traffic Sampling and Forwarding Configuration,” explains how to configure traffic sampling and forwarding, including specifying traffic threshold value, sampling rate, and run length for traffic sampling, as well as load-balancing policy, and configuring DNS and TFTP packet forwarding. This chapter also includes sampling and forwarding configuration examples.

  - Chapter 15, “Summary of Traffic Sampling and Forwarding Options Configuration Statements,” explains the traffic sampling configuration statements.

This manual also contains a glossary, a complete index, and an index of statements and commands.

Part Organization

The parts in this manual typically contain the following chapters:

- Overview—Provides background information about and discusses concepts related to the software component described in that part of the book.
- Configuration statements—Lists all the configuration statements available to configure the software component. This list is designed to provide an overview of the configuration statement hierarchy for that software component.
- Configuration guidelines—Describes how to configure all the features of the software component. The first section of the configuration guidelines describes the minimum configuration for that component, listing the configuration statements you must include to enable the software component on the router with only the bare minimum functionality. The remaining sections in the configuration guidelines are generally arranged so that the most common features are near the beginning.
- Statement summary—A reference that lists all configuration statements alphabetically and explains each statement and all its options. The explanation of each configuration statement consists of the following parts:
  - Syntax—Describes the full syntax of the configuration statement. For an explanation of how to read the syntax statements, see “Documentation Conventions” on page xix.
  - Hierarchy level—Tells where in the configuration statement hierarchy you include the statement.
  - Description—Describes the function of the configuration statement.
  - Options—Describes the configuration statement’s options, if there are any. For options with numeric values, the allowed range and default value, if any, are listed. For multiple options, if one option is the default, that fact is stated. If a configuration statement is at the top of a hierarchy of options that are other configuration statements, these options are generally explained separately in the statement summary section.
  - Usage guidelines—Points to the section or sections in the configuration guidelines section that describe how to use the configuration statement.
  - Required privilege level—Indicates the permissions that the user must have to view or modify the statement in the router configuration. For an explanation of the permissions, see the *JUNOS Internet Software Configuration Guide: Getting Started*.
  - See also—Indicates other configuration statements that might provide related or similar functionality.

Using the Indexes

This manual contains two indexes: a complete index, which contains all index entries, and an index that contains only statements and commands.

In the complete index, bold page numbers point to pages in the statement summary chapters. The index entry for each configuration statement always contains at least two entries. The first, with a bold page number on the same line as the statement name, references the statement summary section. The second entry, "usage guidelines," references the section in a configuration guidelines chapter that describes how to use the statement.

Documentation Conventions

General Conventions

This manual uses the following text conventions:

- Statements, commands, filenames, directory names, IP addresses, and configuration hierarchy levels are shown in a sans serif font. In the following example, stub is a statement name and [edit protocols ospf area *area-id*] is a configuration hierarchy level:

  To configure a stub area, include the stub statement at the [edit protocols ospf area *area-id*] hierarchy level:

- In examples, text that you type literally is shown in bold. In the following example, you type the word *show*:

  [edit protocols ospf area *area-id*]
  cli# **show**
  stub <default-metric *metric*>

- Examples of command output are generally shown in a fixed-width font to preserve the column alignment. For example:

```
> show interfaces terse
Interface       Admin Link Proto Local            Remote
at-1/3/0        up    up
at-1/3/0.0      up    up   inet  1.0.0.1          --> 1.0.0.2
                           iso
fxp0            up    up
fxp0.0          up    up   inet  192.168.5.59/24
```

Conventions for Software Commands and Statements

When describing the JUNOS software, this manual uses the following type and presentation conventions:

- Statement or command names that you type literally are shown nonitalicized. In the following example, the statement name is area:

  You configure all these routers by including the following area statement at the [edit protocols ospf] hierarchy level:

- Options, which are variable terms for which you substitute appropriate values, are shown in italics. In the following example, *area-id* is an option. When you type the area statement, you substitute a value for *area-id*.

  area *area-id*;

- Optional portions of a configuration statement are enclosed in angle brackets. In the following example, the "default-metric *metric*" portion of the statement is optional:

  stub <default-metric *metric*>;

- For text strings separated by a pipe (|), you must specify either *string1* or *string2*, but you cannot specify both or neither of them. Parentheses are sometimes used to group the strings.

  string1 | *string2*
  (*string1* | *string2*)

  In the following example, you must specify either broadcast or multicast, but you cannot specify both:

  broadcast | multicast

- For some statements, you can specify a set of values. The set must be enclosed in square brackets. For example:

  community *name* members [*community-ids*]

- The configuration examples in this manual are generally formatted in the way that they appear when you issue a show command. This format includes braces ({ }) and semicolons. When you type configuration statements in the CLI, you do not type the braces and semicolons. However, when you type configuration statements in an ASCII file, you must include the braces and semicolons. For example:

```
[edit]
cli# set routing-options static route default nexthop address retain
[edit]
cli# show
routing-options {
    static {
        route default {
            nexthop address;
            retain;
        }
    }
}
```

- Comments in the configuration examples are shown either preceding the lines that the comments apply to, or more often, they appear on the same line. When comments appear on the same line, they are preceded by a pound sign (#) to indicate where the comment starts. In an actual configuration, comments can only precede a line; they cannot be on the same line as a configuration statement. For example:

```
protocols {
    mpls {
        interface (interface-name | all);    # Required to enable MPLS on the interface
    }
    rsvp {                                   # Required for dynamic MPLS only
        interface interface-name;
    }
}
```

- The general syntax descriptions provide no indication of the number of times you can specify a statement, option, or keyword. This information is provided in the text of the statement summary.

List of Technical Publications

Table 1 lists the software and hardware books for Juniper Networks routers and describes the contents of each book.

Table 1: Juniper Networks Technical Documentation

Book	Description
JUNOS Internet Software Configuration Guides	
Getting Started	Provides an overview of the JUNOS Internet software and describes how to install and upgrade the software. This manual also describes how to configure system management functions and how to configure the chassis, including user accounts, passwords, and redundancy.
Interfaces and Class of Service	Provides an overview of the interface and class-of-service functions of the JUNOS Internet software and describes how to configure the interfaces on the router.
MPLS Applications	Provides an overview of traffic engineering concepts and describes how to configure traffic engineering protocols.
Multicast	Provides an overview of multicast concepts and describes how to configure multicast routing protocols.
Network Management	Provides an overview of network management concepts and describes how to configure various network management features, such as SNMP, accounting options, and cflowd.
Policy Framework	Provides an overview of policy concepts and describes how to configure routing policy, firewall filters, and forwarding options.
Routing and Routing Protocols	Provides an overview of routing concepts and describes how to configure routing, routing instances, and unicast routing protocols.
VPNs	Provides an overview of Layer 2 and Layer 3 Virtual Private Networks (VPNs), describes how to configure VPNs, and provides configuration examples.
JUNOS Internet Software References	
Operational Mode Command Reference: Interfaces	Describes the JUNOS Internet software operational mode commands you use to monitor and troubleshoot Juniper Networks routers.
Operational Mode Command Reference: Protocols, Class of Service, Chassis, and Management	Describes the JUNOS Internet software operational mode commands you use to monitor and troubleshoot Juniper Networks routers.
System Log Messages Reference	Describes how to access and interpret system log messages generated by JUNOS software modules and provides a reference page for each message.
JUNOScript API Documentation	
JUNOScript API Guide	Describes how to use the JUNOScript API to monitor and configure Juniper Networks routers.
JUNOScript API Reference	Provides a reference page for each tag in the JUNOScript API.
JUNOS Internet Software Comprehensive Index	
Comprehensive Index	Provides a complete index of all JUNOS Internet software books and the *JUNOScript API Guide*.
Hardware Documentation	
Hardware Guide	Describes how to install, maintain, and troubleshoot routers and router components. Each router platform (M5 and M10 routers, M20 router, M40 router, M40e router, M160 router, and T640 routing node) has its own hardware guide.
PIC Guide	Describes the router Physical Interface Cards (PICs). Each router platform has its own PIC guide.

Documentation Feedback

We are always interested in hearing from our customers. Please let us know what you like and do not like about the Juniper Networks documentation, and let us know of any suggestions you have for improving the documentation. Also, let us know if you find any mistakes in the documentation. Send your feedback to tech-doc@juniper.net.

How to Request Support

For technical support, contact Juniper Networks at support@juniper.net, or at 1-888-314-JTAC (within the United States) or 408-745-2121 (from outside the United States).

Part 1

Policy Framework

■ Policy Framework Overview on page 3

Chapter 1 Policy Framework Overview

The JUNOS Internet software provides a *policy framework*, which is a collection of JUNOS policies that allows you to control flows of routing information and packets. The policy framework is composed of the following policies:

- Routing policy—Allows you to control the routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table
- Firewall filter policy—Allows you to control packets transiting the router to a network destination and packets destined for and sent by the router

The term *firewall filter policy* is used here to emphasize that a firewall filter is a policy and shares some fundamental similarities with a routing policy. However, when referring to a firewall filter policy in the rest of this manual, the term *firewall filter* is used.

This chapter discusses the following topics related to understanding the JUNOS policy framework:

Router Flows Affected by Policies

The JUNOS policies affect the following router flows:

- Flow of routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table. The Routing Engine handles this flow. *Routing information* is the information about routes learned by the routing protocols from a router's neighbors. This information is stored in routing tables and is subsequently advertised by the routing protocols to the router's neighbors. Routing policies allow you to control the flow of this information.

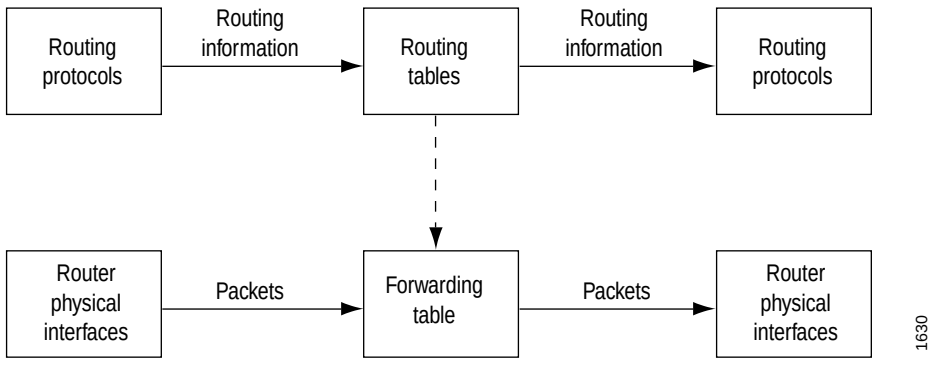
- Flow of data packets in and out of the router physical interfaces. The Packet Forwarding Engine handles this flow. *Data packets* are chunks of data that transit the router as they are being forwarded from a source to a destination. When a router receives a data packet on an interface, the router determines where to forward the packet by looking in the forwarding table for the best route to a destination. The router then forwards the data packet toward its destination through the appropriate interface. Firewall filters allow you to control the flow of these data packets.

- Flow of local packets from the router physical interfaces and to the Routing Engine. The Routing Engine handles this flow. *Local packets* are chunks of data that are destined for or sent by the router. Local packets usually contain routing protocol data, data for IP services such as telnet or secure shell (ssh), and data for administrative protocols such as the Internet Control Message Protocol (ICMP). When the Routing Engine receives a local packet, it forwards the packet to the appropriate daemon or to the kernel, which are both part of the Routing Engine, or to the Packet Forwarding Engine. Firewall filters allow you to control the flow of these local packets.

In the rest of this chapter, the term *packets* refers to both data and local packets unless explicitly stated otherwise.

Figure 1 illustrates the flows through the router. Although the flows are very different from each other, they are also interdependent. Routing policies determine which routes are placed in the forwarding table. The forwarding table, in turn, has an integral role in determining the appropriate physical interface through which to forward a packet.

Figure 1: Flows of Routing Information and Packets

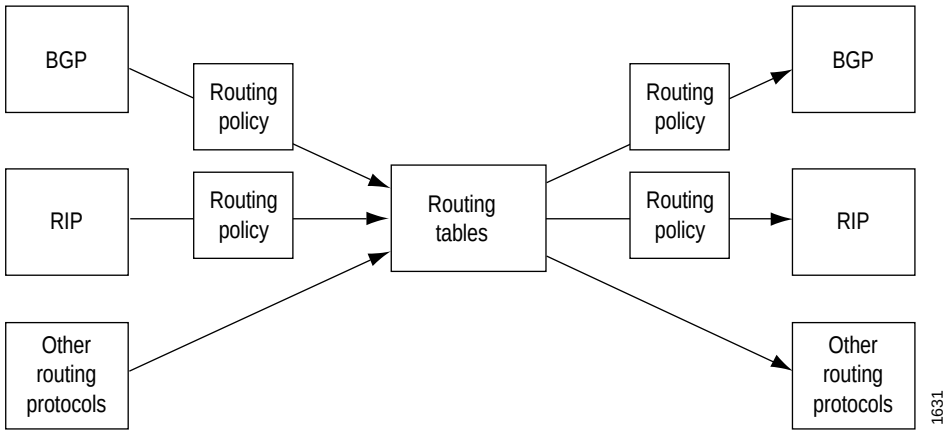


You can configure routing policies to control which routes the routing protocols place in the routing tables and to control which routes the routing protocols advertise from the routing tables (see Figure 2). The routing protocols advertise active routes only from the routing tables. (An *active route* is a route that is chosen from all routes in the routing table to reach a destination. For information about the active route selection process, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.)

You can also use routing policies to do the following:

- Change specific route characteristics, which allow you to control which route is selected as the active route to reach a destination. In general, the active route is also advertised to a router's neighbors.
- Change to the default Border Gateway Protocol (BGP) route flap-damping values.
- Perform per-packet load balancing.
- Enable class of service (CoS).

Figure 2: Routing Policies to Control Routing Information Flow

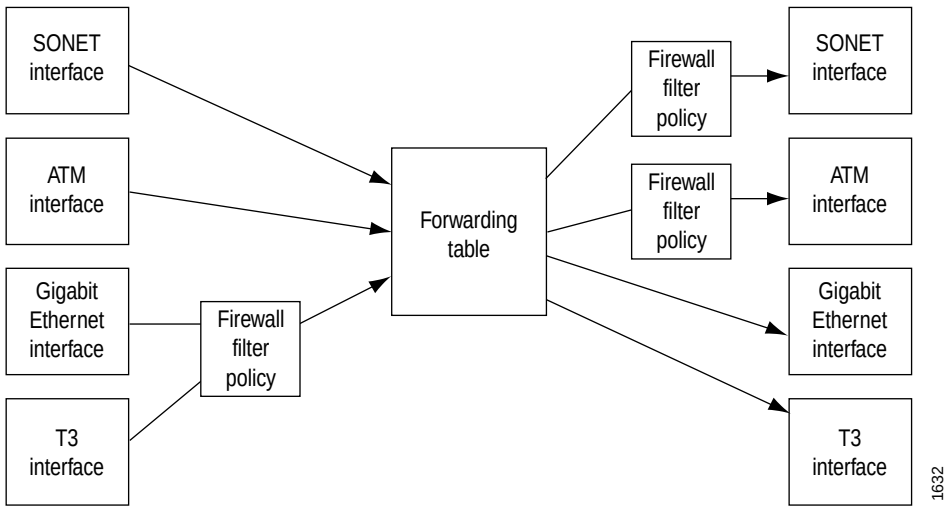


You can configure firewall filters to control the following (see Figure 3):

- Which data packets are accepted on and transmitted from the physical interfaces. To control the flow of data packets, you apply firewall filters to the physical interfaces.
- Which local packets are transmitted from the physical interfaces and to the Routing Engine. To control local packets, you apply firewall filters on the loopback interface, which is the interface to the Routing Engine.

Firewall filters provide a means of protecting your router from excessive traffic transiting the router to a network destination or destined for the Routing Engine. Firewall filters that control local packets can also protect your router from external aggressions such as denial-of-service attacks.

Figure 3: Firewall Filters to Control Packet Flow



Policy Architecture

A *policy* is a mechanism in the JUNOS policy framework that allows you to configure criteria against which something can be compared and an action that is performed if the criteria are met.

All policies in the JUNOS policy framework share the following architecture and configuration fundamentals:

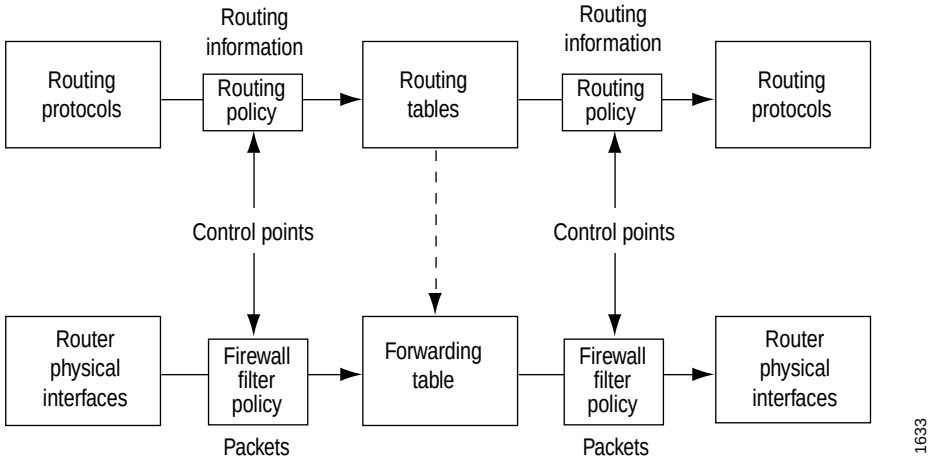
> **Note** This section highlights the fundamental architecture that all policies share. Note, however, that the implementation details of routing policies and firewall filters are very different. For information about these differences, see "Comparison of Routing Policies and Firewall Filters" on page 10.

Control Points

All policies provide two points at which you can control routing information or packets through the router (see Figure 4). These control points allow you to control the following:

- Routing information before and after it is placed in the routing table.
- Data packets before and after a forwarding table lookup.
- Local packets before and after they are received by the Routing Engine. (Figure 4 appears to depict only one control point but because of the bidirectional flow of the local packets, two control points actually exist.)

Figure 4: Policy Control Points



Because there are two control points, you can configure policies that control the routing information or data packets before and after their interaction with their respective tables, and policies that control local packets before and after their interaction with the Routing Engine. *Import routing policies* control the routing information that is placed in the routing tables, while *export routing policies* control the routing information that is advertised from the routing tables. *Input firewall filters* control packets that are received on a router interface, while *output firewall filters* control packets that are transmitted from a router interface.

Policy Components

All policies are composed of the following components that you configure:

- *Match conditions*—Criteria against which a route or packets are compared. You can configure one or more criteria. If all criteria match, one or more actions are applied.
- *Actions*—What happens if all criteria match. You can configure one or more actions.
- *Terms*—Named structures in which match conditions and actions are defined. You can define one or more terms.

For more information about these concepts and how they fit into the context of their respective policies, see “Routing Policy Configuration” on page 21 and “Firewall Filter Components” on page 140.

The policy framework software evaluates each incoming and outgoing route or packet against the match conditions in a term. If the criteria in the match conditions are met, the defined action is taken.

In general, the policy framework software compares the route or packet against the match conditions in the first term in the policy, then goes on to the next term, and so on. (For specific information about when the evaluation process ends for each policy, see “Comparison of Routing Policies and Firewall Filters” on page 10.) Therefore, the order in which you arrange terms in a policy is relevant.

The order of match conditions within a term is not relevant because a route or packet must match all match conditions in a term for an action to be taken.

Default Policies and Actions

If an incoming or outgoing route or packet arrives and there is no explicitly configured policy related to the route or to the interface upon which the packet arrives, the action specified by the default policy is taken. A *default policy* is a rule or a set of rules that determine if the route is placed in or advertised from the routing table, or if the packet is accepted into or transmitted from the router interface.

All policies also have default actions in case one of the following situations arises during policy evaluation:

- A policy does not specify a match condition.
- A match occurs, but a policy does not specify an action.
- A match does not occur with a term in a policy and subsequent terms in the same policy exist.
- A match does not occur by the end of a policy.

Configuration Tasks

All policies share a two-step configuration process:

- Define the policy—Define the policy components. The components include criteria against which routes or packets are compared and actions that are performed if the criteria are met. For more information, see “Policy Components” on page 8.
- Apply the policy—Apply the policy to whatever moves the routing information or packets through the router, for example, the routing protocol or the router interface.

A defined policy does not take effect until you apply it.

Policy Configuration Recommendations

The JUNOS policy architecture is simple and straightforward. However, the actual implementation of each policy adds layers of complexity to the policy as well as adding power and flexibility to your router's capabilities. Configuring a policy has a major impact on the flow of routing information or packets within and through the router. For example, you can configure a routing policy that does not allow routes associated with a particular customer to be placed in the routing table. As a result of this routing policy, the customer routes are not used to forward data packets to various destinations and the routes are not advertised by the routing protocol to neighbors.

Before configuring a policy, determine what you want to accomplish with it and thoroughly understand how to achieve your goal using the various match conditions and actions. Also, make certain that you understand the default policies and actions for the policy you are configuring.

Comparison of Routing Policies and Firewall Filters

Although routing policies and firewall filters share an architecture, as described in "Policy Architecture" on page 7, several differences exist. There purposes are different as summarized in Table 2. Therefore the implementation details and, consequently, the configuration methods for two policies are very different. Table 3 of implementation details for routing policies and firewall filters, highlight the similarities and differences.

For complete information about routing policies, see "Routing Policies" on page 13. For complete information about firewall filters, see "Firewall Filters" on page 137.

Table 2: Purpose of Routing Policies and Firewall Filters

Policies	Source	Policy Purpose
Routing policies	Routing information is generated by internal networking peers.	To control the size and content of the routing tables, which routes are advertised, and which routes are considered the best to reach various destinations.
Firewall filters	Packets are generated by internal and external devices through which hostile attacks can be perpetrated.	To protect your router and network from excessive incoming traffic or hostile attacks that can disrupt network service, and to control which packets are forwarded from which router interfaces.

Table 3: Implementation Differences Between Routing Policies and Firewall Filters

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Control points	Control routing information that is placed in the routing table with an import routing policy and advertised from the routing table with an export routing policy.	Control packets that are accepted on a router interface with an input firewall filter and that are forwarded from an interface with an output firewall filter.
Configuration tasks: ■ Define policy ■ Apply policy	Define a policy that contains terms, match conditions, and actions. Apply one or more export or import policies to a routing protocol. You can also apply a *policy expression*, which uses Boolean logical operators with multiple import or export policies. You can also apply one or more export policies to the forwarding table.	Define a policy that contains terms, match conditions, and actions. Apply one input or output firewall filter to a physical interface or physical interface group to filter data packets received by or forwarded to a physical interface (on routers with an Internet Processor II ASIC only). You can also apply one input or output firewall filter to the router's loopback interface, which is the interface to the Routing Engine (on all routers). This allows you to filter local packets received by or forwarded from the Routing Engine.
Terms	Configure as many terms as desired. Define a name for each term. Terms are evaluated in the order in which you specify them. Evaluation of a policy ends after a packet matches the criteria in a term and the defined or default policy action of accept or reject is taken. The route is not evaluated against subsequent terms in the same policy or subsequent policies.	Configure as many terms as desired. Define a name for each term. Terms are evaluated in the order in which you specify them. Evaluation of a firewall filter ends after a packet matches the criteria in a term and the defined or default action is taken. The packet is not evaluated against subsequent terms in the firewall filter.
Match conditions	Specify zero or more criteria that a route must match. You can specify criteria based on source, destination, or properties of a route. You can also specify the following match conditions, which require more configuration: ■ Autonomous system (AS) path expression—A combination of AS numbers and regular expression operators. ■ Community—A group of destinations that share a common property. ■ Prefix list—A named list of prefixes. ■ Route list—A list of destination prefixes. ■ Subroutine—A routing policy that is called repeatedly from other routing policies.	Specify zero or more criteria that a packet must match. You must match various fields in the packet's header. The fields are grouped into the following categories: ■ Numeric values, such as port and protocol numbers. ■ Prefix values, such as IP source and destination prefixes. ■ Bit-field values—Whether particular bits in the fields are set, such as IP options, TCP flags, and IP fragmentation fields. You can specify the fields using Boolean logical operators.

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Actions	Specify zero or one action to take if a route matches all criteria. You can specify the following actions: ■ Accept—Accept the route into the routing table, and propagate it. After this action is taken, the evaluation of subsequent terms and policies ends. ■ Reject—Do not accept the route into the routing table, and do not propagate it. After this action is taken, the evaluation of subsequent terms and policies ends. In addition to the actions described above, you can also specify zero or more of the following types of actions: ■ Next term—Evaluate the next term in the routing policy. ■ Next policy—Evaluate the next routing policy. ■ Actions that manipulate characteristics associated with a route as the routing protocol places it in the routing table or advertises it from the routing table. ■ Trace action, which logs route matches.	Specify zero or one action to take if a packet matches all criteria. (We recommend that you always explicitly configure an action.) You can specify the following actions: ■ Accept—Accept a packet. ■ Discard—Discard a packet silently, without sending an ICMP message. ■ Reject—Discard a packet, and send an ICMP destination unreachable message. ■ Routing instance—Specify a routing table to which packets are forwarded. ■ Next term—Evaluate the next term in the firewall filter. In addition to zero or one of the actions described above, you can also specify zero or more action modifiers. You can specify the following action modifiers: ■ Count—Add packet to a count total. ■ Forwarding class—Set the packet forwarding class to a specified value from 0 through 3. ■ IPSec security association—Used with the source and destination address match conditions, specify an IP Security (IPSec) security association (SA) for the packet. ■ Log—Store the header information of a packet on the Routing Engine. ■ Loss priority—Set the packet loss priority (PLP) bit to a specified value, 0 or 1. ■ Policer—Apply rate-limiting procedures to the traffic. ■ Sample—Sample the packet traffic. ■ Syslog—Log an alert for the packet.
Default policies and actions	If an incoming or outgoing route arrives and a policy related to the route is not explicitly configured, the action specified by the default policy for the associated routing protocol is taken. The following default actions exist for routing policies: ■ If a policy does not specify a match condition, all routes evaluated against the policy match. ■ If a match occurs but the policy does not specify an accept, reject, next term, or next policy action, one of the following occurs: ■ The next term, if present, is evaluated. ■ If no other terms are present, the next policy is evaluated. ■ If no other policies are present, the action specified by the default policy is taken. ■ If a match does not occur with a term in a policy and subsequent terms in the same policy exist, the next term is evaluated. ■ If a match does not occur with any terms in a policy and subsequent policies exist, the next policy is evaluated. ■ If a match does not occur by the end of a policy and no other policies exist, the accept or reject action specified by the default policy is taken.	If an incoming or outgoing packet arrives on an interface and a firewall filter is not configured for the interface, the default policy is taken (the packet is accepted). The following default actions exist for firewall filters: ■ If a firewall filter does not specify a match condition, all packets are considered to match. ■ If a match occurs but the firewall filter does not specify an action, the packet is accepted. ■ If a match occurs, the defined or default action is taken and the evaluation ends. Subsequent terms in the firewall filter are not evaluated, unless the next term action is specified. ■ If a match does not occur with a term in a firewall filter and subsequent terms in the same filter exist, the next term is evaluated. ■ If a match does not occur by the end of a firewall filter, the packet is discarded.

Part 2

Routing Policies

Chapter 2
Routing Policy Framework Overview

All routing protocols store their routing information in routing tables. From these tables, the routing protocols calculate the best route to each destination and place these routes in a forwarding table. These routes are then used to forward routing protocol traffic toward a destination, and they can be advertised to neighbors using one or more routing protocols.

Instead of referring to the multiple routing tables that the JUNOS software maintains, the discussion in the rest of this chapter assumes the inet.0 routing table unless explicitly stated otherwise. By default, the JUNOS software stores unicast Internet Protocol Version 4 (IPv4) routes in the inet.0 routing table. For information about all the routing tables, see "Routing Tables Affected by Routing Policies" on page 18.

In general, the routing protocols place all their routes in the routing table and advertise a limited set of routes from the routing table. The general rules for handling the routing information between the routing protocols and the routing table are known as the *routing policy framework*.

The routing policy framework is composed of default rules for each routing protocol that determine which routes the protocol places in the routing table and advertises from the routing table. The default rules for each routing protocol are known as *default routing policies*.

You can create routing policies to preempt the default policies, which are always present. A *routing policy* is a mechanism in the JUNOS software that allows you to modify the routing policy framework to suit your needs. You can create and implement your own routing policies to do the following:

- Control which routes a routing protocol places in the routing table.
- Control which active routes a routing protocol advertises from the routing table. (An *active route* is a route that is chosen from all routes in the routing table to reach a destination. For information about the active route selection process, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.)
- Manipulate the route characteristics as a routing protocol places it in the routing table or advertises it from the routing table.

You can manipulate the route characteristics to control which route is selected as the active route to reach a destination. The active route is placed in the forwarding table and used to forward traffic toward the route's destination. In general, the active route is also advertised to a router's neighbors.

To create a routing policy, you must define the policy and apply it. You define the policy by specifying the criteria that a route must match and the actions to perform if a match occurs. You then apply the policy to a routing protocol or to the forwarding table.

This chapter discusses the following topics related to understanding and creating routing policies:

Before you create your routing policies, we recommend that you read through this entire section to become familiar with the terminology, concepts, and configuration guidelines.

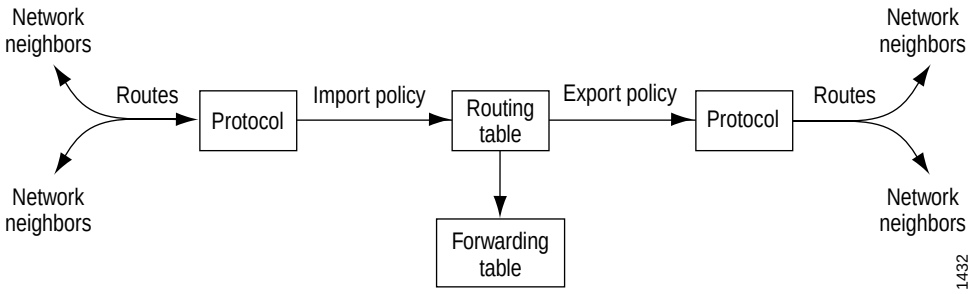
Import and Export

Two terms—*import* and *export*—explain how routes move between the routing protocols and the routing table (see Figure 5):

- When the Routing Engine places the routes of a routing protocol into the routing table, it is *importing* routes into the routing table.
- When the Routing Engine uses active routes from the routing table to send a protocol advertisement, it is *exporting* routes from the routing table.

The process of moving routes between a routing protocol and the routing table is described always *from the point of view of the routing table*. That is, routes are *imported into* a routing table from a routing protocol and they are *exported from* a routing table to a routing protocol. Remember this distinction when working with routing policies.

Figure 5: Importing and Exporting Routes



When evaluating routes for export, the Routing Engine uses only active routes from the routing table. For example, if a routing table contains multiple routes to the same destination and one route has a preferable metric, only that route is evaluated. In other words, an export policy does not evaluate all routes; it evaluates only those routes that a routing protocol is allowed to advertise to a neighbor. For more information about the active path selection algorithm, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

By default, the Border Gateway Protocol (BGP) advertises active routes. However, you can configure BGP to advertise *inactive routes*, which go to the same destination as other routes but have less preferable metrics. For more information about advertising inactive routes, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

Table 4 lists the routing protocols from which the routing table can import routes and to which the routing table can export routes. Table 4 also lists direct and explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. (An *explicitly configured route* is a route that you have configured. *Direct routes* are not explicitly configured; they are created as a result of IP addresses being configured on an interface.) Explicitly configured routes include aggregate, generated, local, and static routes. (An *aggregate route* is a route that distills groups of routes with common addresses into one route. A *generated route* is a route used when the routing table has no information about how to reach a particular destination. A *local route* is an IP address assigned to a router interface. A s*tatic route* is a nonchanging route to a destination.)

The policy framework software treats direct and explicitly configured routes as if they are learned through routing protocols; therefore, they can be imported into the routing table. Routes cannot be exported from the routing table to the pseudoprotocol, because this protocol is not a real routing protocol. However, aggregate, direct, generated, and static routes can be exported from the routing table to routing protocols, whereas local routes cannot.

For information about the default routing policies for each routing protocol, see Table 6 on page 19. For information about the import and export routing policies supported for each routing protocol and the level at which you can apply these policies, see Table 8 on page 26.

Table 4: Protocols That Can Be Imported to and Exported from the Routing Table

Protocol	Import	Export
Border Gateway Protocol (BGP)	Yes	Yes
Distance Vector Multicast Routing Protocol (DVMRP)	Yes	Yes
Intermediate System-to-Intermediate System (IS-IS)	Yes	Yes
Label Distribution Protocol (LDP)	Yes	Yes
Multiprotocol Label Switching (MPLS)	Yes	No
Open Shortest Path First (OSPF)	Yes	Yes
Protocol Independent Multicast (PIM) dense mode	Yes	Yes
PIM sparse mode	Yes	Yes
PIM sparse-dense mode	Yes	Yes
Pseudoprotocol: ■ Direct routes ■ Explicitly configured routes ■ Aggregate routes ■ Generated routes ■ Local routes ■ Static routes	Yes	No
Routing Information Protocol (RIP) and Routing Information Protocol Next-Generation (RIPng)	Yes	Yes

Routing Tables Affected by Routing Policies

Table 5 lists the routing tables affected by default and user-defined routing policies and the types of routes that each routing table stores.

Table 5: Routing Tables Affected by Routing Policies

Routing Table	Type of Routes Stored
inet.0	Unicast IPv4 routes
instance-name.inet.0	Unicast IPv4 routes for a particular routing instance
inet.1	Multicast IPv4 routes
inet.2	Unicast IPv4 routes for multicast reverse path forwarding (RPF) lookup
inet.3	MPLS routes
mpls.0	MPLS routes for label-switched path (LSP) next hops
inet6.0	Unicast Internet Protocol Version 6 (IPv6) routes


Note

The discussion in the rest of this chapter assumes the inet.0 routing table unless explicitly stated otherwise.

For more information about routing tables, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols.*

Default Routing Policies and Actions

You must be familiar with the default routing policies to know when you need to modify them to suit your needs. Table 6 summarizes the default routing policies for each routing protocol that imports and exports routes. The actions in the default routing policies are taken if you have not explicitly configured a routing policy. This table also shows direct and explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. Explicitly configured routes include aggregate, generated, and static routes.

The default import policy is always the same: accept all routes learned from the protocol. Table 6 includes information about the routing tables used by each protocol.

Table 6: Default Routing Policies

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
BGP	Accept all BGP IPv4 routes learned from configured neighbors and import into the inet.0 routing table. Accept all BGP IPv6 routes learned from configured neighbors and import into the inet6.0 routing table.	Accept and export active BGP routes.
DVMRP	Accept all DVMRP routes and import into the inet.1 routing table.	Accept and export active DVMRP routes.
IS-IS	Accept all IS-IS routes and import into the inet.0 and inet6.0 routing tables. (You cannot override or change this default policy.)	Reject everything. (The protocol uses flooding to announce local routes and any learned routes.)
LDP	Accept all LDP routes and import into the inet.3 routing table.	Accept and export active LDP routes.
MPLS	Accept all MPLS routes and import into the inet.3 routing table.	Accept and export active MPLS routes.
OSPF	Accept all OSPF routes and import into the inet.0 routing table. (You cannot override or change this default policy.)	Reject everything. (The protocol uses flooding to announce local routes and any learned routes.)
PIM dense mode	Accept all PIM dense mode routes and import into the inet.1 routing table.	Accept active PIM dense mode routes.
PIM sparse mode	Accept all PIM sparse mode routes and import into the inet.1 routing table.	Accept and export active PIM sparse mode routes.
Pseudoprotocol: ■ Direct routes ■ Explicitly configured routes: ■ Aggregate routes ■ Generated routes ■ Static routes	Accept all direct and explicitly configured routes and import into the inet.0 routing table.	The pseudoprotocol cannot export any routes from the routing table because it is not a routing protocol. Routing protocols can export these or any routes from the routing table.
RIP	Accept all RIP routes learned from configured neighbors and import into the inet.0 routing table.	Reject everything. To export RIP routes, you must configure an export policy for RIP.
RIPng	Accept all RIPng routes learned from configured neighbors and import into the inet6.0 routing table.	Reject everything. To export RIPng routes, you must configure an export policy for RIPng.

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
Test policy	Accept all routes. For additional information about test policy, see "Routing Policy Tests" on page 32.	

When multiple routes for a destination exist in the routing table, the protocol selects an active route and that route is placed in the appropriate routing table. For equal-cost routes, the JUNOS software places multiple next hops in the appropriate routing table.

When a protocol is exporting routes from the routing table, it exports active routes only. This applies to actions specified by both default and user-defined export policies.

You cannot change the default import policy for the link-state protocols IS-IS and OSPF. As link-state protocols, IS-IS and OSPF exchange routes between systems within an autonomous system (AS). All routers and systems within an AS must share the same link-state database, which includes routes to reachable prefixes and the metrics associated with the prefixes. If an import policy is configured and applied to IS-IS or OSPF, some routes might not be learned or advertised or the metrics for learned routes might be altered, which would make a consistent link-state database impossible.

The default export policy for IS-IS and OSPF protocols is to reject everything. These protocols do not actually export their internally learned routes (the directly connected routes on interfaces that are running the protocol). Both IS-IS and OSPF protocols use a procedure called *flooding* to announce local routes and any routes learned by the protocol. The flooding procedure is internal to the protocol, and is unaffected by the policy framework. Exporting can be used only to announce information from other protocols, and the default is not to do so.

For information about the routing protocols from which the routing table can import routes and to which routing protocols the routing table can export routes, see Table 4 on page 18. For information about the user-defined import and export policies supported for each routing protocol and the level at which you can apply these policies, see Table 8 on page 26.

The following default actions are taken if the following situations arise during policy evaluation:

- If a policy does not specify a match condition, all routes evaluated against the policy match.
- If a match occurs but the policy does not specify an accept, reject, next term, or next policy action, one of the following occurs:
  - The next term, if present, is evaluated.
  - If no other terms are present, the next policy is evaluated.
  - If no other policies are present, the action specified by the default policy is taken.
- If a match does not occur with a term in a policy and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the accept or reject action specified by the default policy is taken.

When to Create Routing Policies

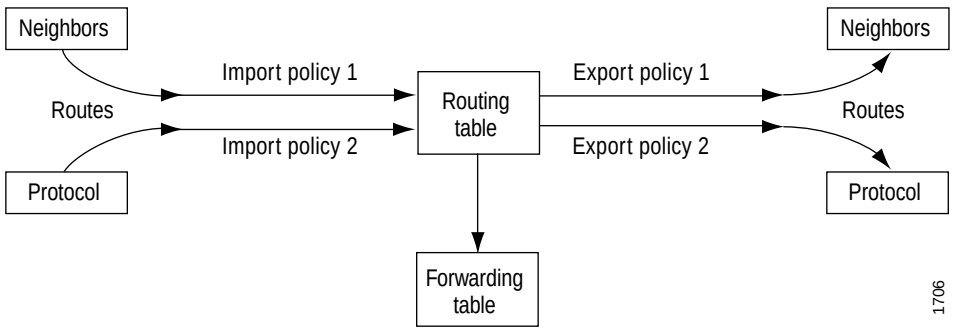
The following are typical circumstances under which you might want to preempt the default routing policies in the routing policy framework by creating your own routing policies:

- You do not want a protocol to import all routes into the routing table. If the routing table does not learn about certain routes, they can never be used to forward packets and they can never be redistributed into other routing protocols.
- You do not want a routing protocol to export all the active routes it learns.
- You want a routing protocol to announce active routes learned from another routing protocol, which is sometimes called *route redistribution*.
- You want to manipulate route characteristics, such as the preference value, AS path, or community. You can manipulate the route characteristics to control which route is selected as the active route to reach a destination. In general, the active route is also advertised to a router's neighbors.
- You want to change the default BGP route flap-damping parameters.
- You want to perform per-packet load balancing.
- You want to enable class of service (CoS).

Routing Policy Configuration

As shown in Figure 6, you use *import routing policies* to control which routes routing protocols place in the routing table, and *export routing policies* to control which routes a routing protocol advertises from the routing table to its neighbors.

Figure 6: Import and Export Routing Policies

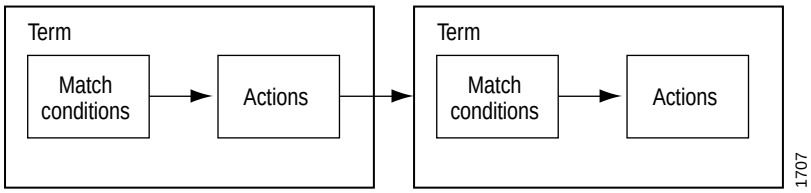


To create a routing policy, you must define the following components:

- *Match conditions*—Criteria that a route must match. If a route matches all of the criteria, one or more actions are applied to the route.
- *Actions*—What to do if a route matches. The actions can specify whether to accept or reject the route, control how a series of policies is evaluated, and manipulate the characteristics associated with a route. You can configure one or more actions.

You typically define match conditions and actions within a *term*. Figure 7 shows the routing policy components, including the term.

Figure 7: Routing Policy Components



After defining a routing policy, you then apply it to a routing protocol or to the forwarding table.

This section provides more information about creating routing policies:

- Match Conditions on page 22
- Named Match Conditions on page 23
- Actions on page 24
- Terms on page 24
- Routing Policy Application on page 25

Match Conditions

A *match condition* defines the criteria that a route must match. You can define one or more match conditions. If a route matches all match conditions, one or more actions are applied to the route.

Match conditions fall into two categories: standard and extended. In general, the extended match conditions include criteria that are defined separately from the routing policy (AS path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. For more information about them, see “Configure Extended Match Conditions” on page 83. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions, also called Named Match Conditions.

Table 7 describes each match condition, including its category, when you typically use it, and any relevant notes about it. For more information about match conditions, see Table 9 on page 41.

Table 7: Match Conditions

Match Condition	Category	When to Use	Notes
AS path regular expression—A combination of AS numbers and regular expression operators.	Extended	(BGP only) Match a route based on its AS path. (An AS path consists of the AS numbers of all routers a packet must go through to reach a destination.) You can specify an exact match with a particular AS path or a less precise match.	You use regular expressions to match the AS path.
Community—A group of destinations that share a property. (Community information is included as a path attribute in BGP update messages.)	Extended	Match a group of destinations that share a property. Use a routing policy to define a community that specifies a group of destinations you want to match and one or more actions that you want taken on this community.	Actions can be performed on the entire group. You can create multiple communities associated with a particular destination. You can create match conditions using regular expressions.
Prefix list—A named list of IP addresses.	Extended	Match a route based on prefix information. You can specify an exact match of a particular route only.	You can specify a common action only for all prefixes in the list.
Route list—A list of destination prefixes.	Extended	Match a route based on prefix information. You can specify an exact match of a particular route or a less precise match.	You can specify an action for each prefix in the route list or a common action for all prefixes in the route list.
Standard—A collection of criteria that can match a route.	Standard	Match a route based on one of the following criteria: area ID, color, external route, family, instance (routing), interface name, level number, local preference, metric, neighbor address, next-hop address, origin, preference, protocol, routing table name, or tag. For the protocol criterion, you can specify one of the following: BGP, direct, DVMRP, IS-IS, local, MPLS, OSPF, PIM dense mode, PIM sparse mode, RIP, RIPng, static, and aggregate.	None.
Subroutine—A routing policy that is called repeatedly from another routing policy.	Extended	Use an effective routing policy in other routing policies. You can create a subroutine that you can call over and over from other routing policies.	The subroutine action influences but does not necessarily determine the final action. For more information, see “How a Routing Policy Subroutine Is Evaluated” on page 30.

Named Match Conditions

Some match conditions are defined separately from the routing policy and are given names. You then reference the name of the match condition in the definition of the routing policy itself. Named match conditions allow you to do the following:

- Reuse match conditions in other routing policies.
- Read configurations that include complex match conditions more easily.

Named match conditions include communities, prefix lists, and AS path regular expressions. For more information about these match conditions, see Table 7 on page 23.

Actions

An *action* is what the policy framework software does if a route matches all criteria defined in a match condition. You can configure one or more actions in a term. The policy framework software supports the following types of actions:

- Flow control actions, which affect whether to accept or reject the route or whether to evaluate the next term or routing policy
- Actions that manipulate route characteristics
- Trace action, which logs route matches

Manipulating the route characteristics allows you to control which route is selected as the active route to reach a destination. In general, the active route is also advertised to a router's neighbors. You can manipulate the following route characteristics: AS path, class, color, community, damping parameters, destination class, external type, next hop, load balance, local preference, metric, origin, preference, and tag.

For the numeric information (color, local preference, metric, preference, and tag), you can set a specific value or change the value by adding or subtracting a specified amount. The addition and subtraction operations do not allow the value to exceed a maximum value and drop below a minimum value.

For more information about the properties you can change and the addition and subtraction operations, see Table 11 on page 45.

Terms

A *term* is a named structure in which match conditions and actions are defined. You can define one or more terms.

In general, the policy framework software compares a route against the match conditions in the first term in the first routing policy, then goes on to the next term and the next policy if present, and so on, until an explicitly configured or default action of accept or reject is taken. Therefore, the order in which you arrange terms in a policy is relevant.

The order of match conditions in a term is not relevant since a route must match all match conditions in a term for an action to be taken.

Routing Policy Application

After defining a routing policy, as discussed in "Match Conditions" on page 22 and "Actions" on page 24, you can apply it to one of the following:

- Routing protocols—BGP, DVMRP, IS-IS, LDP, MPLS, OSPF, PIM dense mode, PIM sparse mode, PIM sparse-dense mode, RIP, and RIPng
- Pseudoprotocol—Explicitly created routes, which include aggregate and generated routes
- Forwarding table

For information about applying routing policies to routing protocols and the pseudoprotocol, see "Routing Protocols" on page 25. For information about applying routing policies to the forwarding table, see "Forwarding Table" on page 26.

Routing Protocols

When applying routing policies to routing protocols, you must know whether each protocol supports import and export policies and the level at which you can apply these policies. Table 8 summarizes the import and export policy support for each routing protocol. Table 8 also lists explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. Explicitly configured routes include aggregate and generated routes.

You can apply an import policy to aggregate and generated routes, but you cannot apply an export policy to these routes. These routes cannot be exported from the routing table to the pseudoprotocol, because this protocol is not a real routing protocol. However, aggregate and generated routes can be exported from the routing table to routing protocols.

You cannot apply import policies to the link-state protocols IS-IS and OSPF. As link-state protocols, IS-IS and OSPF exchange routes between systems within an AS. All routers and systems within an AS must share the same link-state database, which includes routes to reachable prefixes and the metrics associated with the prefixes. If an import policy is configured and applied to IS-IS or OSPF, some routes might not be learned or advertised or the metrics for learned routes might be altered, which would make a consistent link-state database impossible.

For BGP only, you can also apply import and export policies at group and peer levels as well as at the global level. A peer import or export policy overrides a group import or export policy. A group import or export policy overrides a global import or export policy.

For example, if you define an import policy for an individual peer at the peer level and also define an import policy for the group to which it belongs, the import policy defined for the peer level only is invoked. The group import policy is not used for that peer, but it is applied to other peers in that group.

For RIP and RIPng only, you can apply import policies at the global and neighbor levels and export policies at a group level. For more information about RIP and RIPng, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols* and the *JUNOS Internet Software Configuration Guide: IPv6*.

For information about the routing protocols from which the routing table can import routes and to which routing protocols the routing table can export routes, see Table 4 on page 18. For information about the default routing policies for each routing protocol, see Table 6 on page 19.

Table 8: Apply Routing Policies to Protocols

Protocol	Import Policy	Export Policy	Supported Levels
BGP	Yes	Yes	Import: global, group, peer Export: global, group, peer
DVMRP	Yes	Yes	Global
IS-IS	No	Yes	Export: global
LDP	Yes	Yes	Global
MPLS	No	No	–
OSPF	No	Yes	Export: global
PIM dense mode	Yes	Yes	Global
PIM sparse mode	Yes	Yes	Global
Pseudoprotocol—Explicitly configured routes, which include the following: ■ Aggregate routes ■ Generated routes	Yes	No	Import: global
RIP and RIPng	Yes	Yes	Import: global, neighbor Export: group

Routing Policy Application to Routing Protocols

You can apply the following routing policy elements to a routing protocol:

- Routing policy—You can apply a single routing policy to a routing protocol.
- Chain of routing policies—You can apply multiple routing policies (chains) to a routing protocol.
- Policy expression—You can apply a policy expression to a routing protocol. A *policy expression* uses Boolean logical operators with a routing policy and routing policy chains. The logical operators establish rules by which the policy or chains are evaluated.

Forwarding Table

You can apply export policies to routes being exported from the routing table into the forwarding table for the following features:

- Per-packet load balancing
- Class of service (CoS)

For more information about per-packet load balancing, see “Configure Load-Balance Per-Packet Action” on page 125. For more information about CoS, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

Routing Policy Evaluation

This section provides information about how routing policies are evaluated. It discusses the following topics:

For specific information about how the various match conditions are evaluated, see “Match Conditions” on page 39 and “Configure Extended Match Conditions” on page 83.

How a Routing Policy Is Evaluated

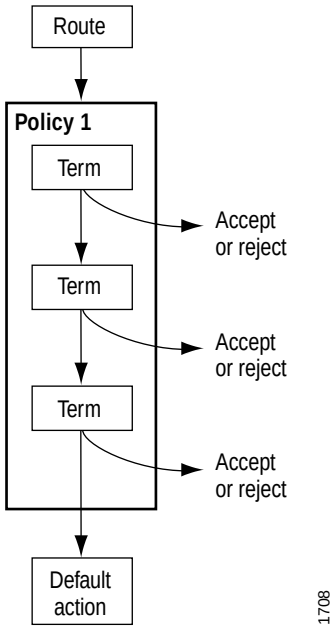
Figure 8 shows how a single routing policy is evaluated. This routing policy consists of multiple terms. Each term consists of match conditions and actions to apply to matching routes. Each route is evaluated against the policy as follows:

1. The route is evaluated against the first term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues as described in Step 2. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.

2. The route is evaluated against the second term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues in a similar manner against the last term. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.

3. If the route matches no terms in the routing policy or the next policy action is specified, the accept or reject action specified by the default policy is taken. For more information about the default routing policies, see “Default Routing Policies and Actions” on page 19.

Figure 8: Routing Policy Evaluation



How a Routing Policy Chain Is Evaluated

Figure 9 shows how a chain of routing policies is evaluated. These routing policies consist of multiple terms. Each term consists of match conditions and actions to apply to matching routes. Each route is evaluated against the policies as follows:

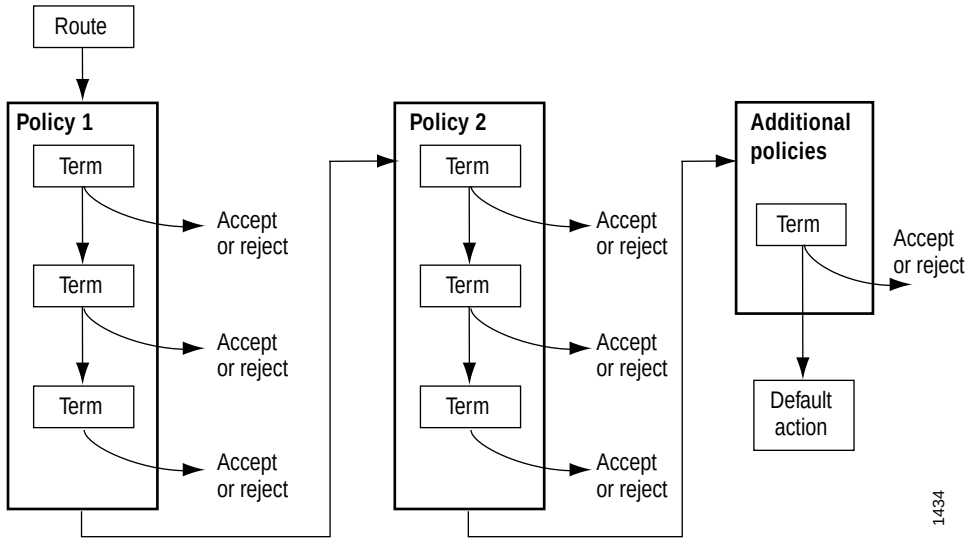
1. The route is evaluated against the first term in the first routing policy. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues as described in Step 2. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.

2. The route is evaluated against the second term in the first routing policy. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues in a similar manner against the last term in the first routing policy. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.

3. If the route does not match a term or matches a term with a next policy action in the first routing policy, it is evaluated against the first term in the second routing policy.

4. The evaluation continues until the route matches a term with an accept or reject action defined or until there are no more routing policies to evaluate. If there are no more routing policies, then the accept or reject action specified by the default policy is taken. For more information about the default routing policies, see “Default Routing Policies and Actions” on page 19.

Figure 9: Routing Policy Chain Evaluation



How a Routing Policy Expression Is Evaluated

To understand how a policy expression is evaluated, you must first understand the Boolean logical operators and the associated logic used in evaluating a policy expression. For more information about policy expressions, including how they are evaluated, see “Apply Policy Expressions” on page 53.

How a Routing Policy Subroutine Is Evaluated

Figure 10 shows how a subroutine is evaluated. The subroutine is included in the first term of the first routing policy in a chain. Each route is evaluated against the subroutine as follows:

1. The route is evaluated against the first term in the first routing policy. If the route does not match all match conditions specified before the subroutine, the subroutine is skipped and the next term in the routing policy is evaluated (see Step 2). If the route matches all match conditions specified before the subroutine, the route is evaluated against the subroutine. If the route matches the match conditions in any of the subroutine terms, two levels of evaluation occur in the following order:

   a. The actions in the subroutine term are evaluated. If one of the actions is accept, evaluation of the subroutine ends and a Boolean value of TRUE is returned to the calling policy. If one of the actions is reject, evaluation of the subroutine ends and FALSE is returned to the calling policy. If one of the actions is meant to manipulate route characteristics, the characteristic is changed regardless of whether accept, reject, or neither action is specified.

      If the subroutine does not specify the accept or reject actions, it uses the accept or reject action specified by the default policy and the values of TRUE or FALSE are returned to the calling policy as described above. (For information about what happens if a termination action is not specified in the term, see "Termination Actions" on page 115. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 19.)

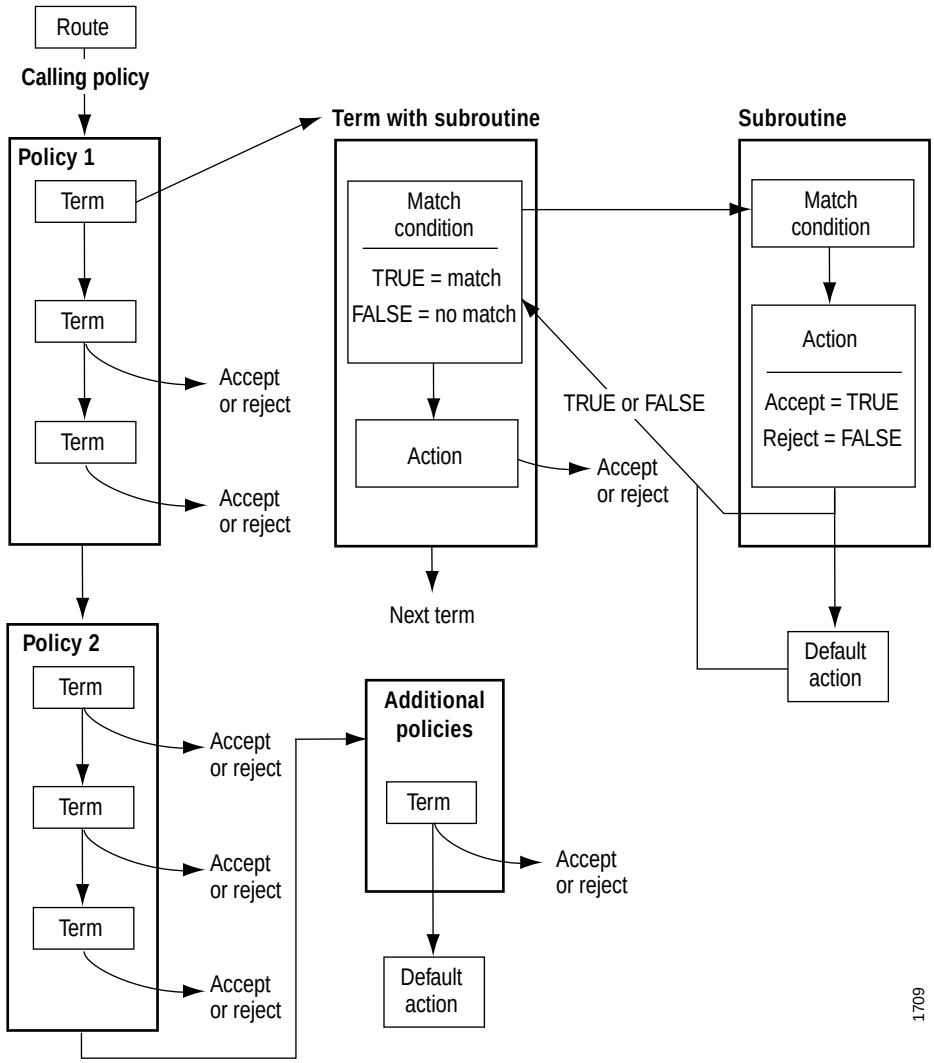
   b. The calling policy's subroutine match condition is evaluated. During this part of the evaluation, TRUE equals a match and FALSE equals no match. If the subroutine returns TRUE to the calling policy, then the evaluation of the calling policy continues. If the subroutine returns FALSE to the calling policy, then the evaluation of the current term ends and the next term is evaluated.

2. The route is evaluated against the second term in the first routing policy. For information about how the subsequent terms and policies are evaluated, see "How a Routing Policy Chain Is Evaluated" on page 28.

If a term defines multiple match conditions, including a subroutine, and a route does not match a condition specified before the subroutine, the evaluation of the term ends and the subroutine is not called and evaluated. In this situation, an action specified in the subroutine that manipulates a route's characteristics is not implemented.

If you specify a policy chain as a subroutine, the entire chain acts as a single subroutine. As with other chains, the action specified by the default policy is taken only when the entire chain does not accept or reject a route.

Figure 10: Routing Policy Subroutine Evaluation



Routing Policy Tests

After you have created a routing policy, you can use the test policy command to ensure that the policy produces the results that you expect before applying the policy in a live environment. This command determines whether the routes specified in your routing policy are accepted or rejected. The default action of the test policy command is accept.

The default policy of the test policy command accepts all routes from all protocols. Test output can be misleading when you are evaluating protocol-specific conditions.

For example, if you define a policy for BGP that accepts routes of a specified prefix and apply it to BGP as an export policy, the BGP routes that match the prefix are advertised to the BGP peers. However, if you test the same policy using the test policy command, the test output might indicate that non-BGP routes have been accepted.

Chapter 3

Routing Policy Configuration Statements

To create a routing policy, you can include the following statements in the configuration:

```
[edit]
policy-options {
    as-path name regular-expression;
    as-path-group group-name;
    community name {
        invert-match;
        members [ community-ids ];
    }
    damping name {
        disable;
        half-life minutes;
        max-suppress minutes;
        reuse number;
        suppress number;
    }
    policy-statement policy-name {
        term term-name {
            from {
                family family-name;
                match-conditions;
                policy subroutine-policy-name;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
                policy subroutine-policy-name;
            }
            then actions;
            default-action (accept | reject);
        }
    }
    prefix-list name {
        ip-addresses;
    }
}
protocols {
    protocol-name {
        import [ policy-names ];
        export [ policy-names ];
    }
}
```

This section includes the following minimum configurations:

Minimum Routing Policy Configuration

To define and apply a routing policy, you must include at least the following statements at the [edit policy-options] and [edit protocols] hierarchy levels. At the [edit protocols] hierarchy level, you can define one or more policy names.

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
            }
            then actions;
        }
    prefix-list name {
        ip-addresses;
    }
}
protocols {
    protocol-name {
        import [ policy-names ];
        export [ policy-names ];
    }
}
```

Minimum Routing Policy Chain Configuration

To define and apply a routing policy chain, you must include at least the following statements at the [edit policy-options] and [edit protocols] hierarchy levels. At the [edit protocols] hierarchy level, you can define a chain of policy names that are evaluated in order.

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
            }
            then actions;
        }
    }
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
            }
            then actions;
        }
    }
    prefix-list name {
        ip-addresses;
    }
}
protocols {
    protocol-name {
        import [ policy-names ];
        export [ policy-names ];
    }
}
```

Minimum Subroutine Configuration

To configure a routing policy that calls a subroutine from another routing policy, you must include at least the following statements at the [edit policy-options] and [edit protocols] hierarchy levels. At the [edit protocols] hierarchy level, you can define one or more policy names.

```
[edit]
policy-options {
    policy-statement subroutine-policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
            }
            then actions;
        }
    }
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                policy subroutine-policy-name;
            }
            to {
                policy subroutine-policy-name;
            }
            then actions;
        }
    }
}
protocols {
    protocol-name {
        import [ policy-names ];
        export [ policy-names ];
    }
}
```

Chapter 4

Configure Routing Policy

This chapter describes the following tasks for configuring routing policies and provides the following examples:

Define Routing Policies

To define a routing policy, include one policy-statement statement at the [edit policy-options] hierarchy level:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                policy subroutine-policy-name;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
                policy subroutine-policy-name;
            }
            then actions;
        }
    prefix-list name {
        ip-addresses;
    }
}
```

The following sections explain the components of the policy statement and provide configuration examples:

Routing Policy Name

Each routing policy is introduced by the keyword policy-statement and a name that identifies it:

```
[edit]
policy-options {
    policy-statement policy-name {
        ...
    }
}
```

The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

Each routing policy name must be unique within a configuration.

Terms

A routing policy statement consists of one or more named terms:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            ...
        }
    }
}
```

The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

Each term name must be unique within a routing policy.

A policy statement can include one unnamed term. To configure an unnamed term, omit the term statement when defining match conditions and actions. The unnamed term is always the last term in the policy and cannot be moved.

Although you can use one unnamed term in each policy statement, we recommend that you name all terms.

Match Conditions

Each term can consist of two statements, from and to, that define match conditions:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                policy subroutine-policy-name;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
                policy subroutine-policy-name;
            }
        }
    }
}
```

In the from statement, you define the criteria that an incoming route must match. You can specify one or more match conditions. If you specify more than one, they all must match the route for a match to occur.

The from statement is optional. If you omit it and the to statement, all routes are considered to match.

In export policies, omitting the from statement in a routing policy term might lead to unexpected results. For more information, see “Side Effects of Omitting the “from” Statement from an Export Policy” on page 58.

In the to statement, you define the criteria that an outgoing route must match. You can specify one or more match conditions. If you specify more than one, they all must match the route for a match to occur. You can specify most of the same match conditions in the to statement that you can in the from statement. In most cases, specifying a match condition in the to statement produces the same result as specifying the same match condition in the from statement.

The to statement is optional. If you omit both it and the from statement, all routes are considered to match.

All conditions in the from and to statements must match for the action to be taken. The match conditions are effectively a logical AND operation. Matching in prefix lists and route lists is handled differently. For more information about these match conditions, including how they are evaluated, see "Configure Prefix Lists" on page 102 and "Configure Route Lists" on page 106.

Table 9 describes the match conditions available for matching an incoming or outgoing route. The table indicates whether you can use the match condition in both from and to statements and whether the match condition functions the same or differently when used with both statements.

Table 9 also indicates whether the match condition is standard or extended. In general, the extended match conditions include criteria that are defined separately from the routing policy (autonomous system [AS] path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. For more information about them, see "Configure Extended Match Conditions" on page 83. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions.

For examples of using the from and to statements, see "Examples: Routing Policy Configuration" on page 62.

Table 9: Routing Policy Match Conditions

Match Condition	Match Condition Category	from Statement Description	to Statement Description
area *area-id*	Standard	(Open Shortest Path First [OSPF] only) Area identifier. In a from statement used with an export policy, match a route learned from the specified OSPF area when exporting OSPF routes into other protocols.	
as-path *name*	Extended	(Border Gateway Protocol [BGP] only) Name of an AS path regular expression. For more information, see "Configure AS Path Regular Expressions" on page 83.	
as-path-group *group-name*	Extended	(Border Gateway Protocol [BGP] only) Name of an AS path group regular expression. For more information, see "Configure AS Path Regular Expressions" on page 83.	
color *preference* color2 *preference*	Standard	Color value. You can specify preference values (color and color2) that are finer-grained than those specified in the preference and preference2 match conditions. The color value can be a number in the range 0 through 4,294,967,295 (2^{32} –1), A lower number indicates a more preferred route. For more information about preference values, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.	
community [*names*]	Extended	Name of one or more communities. If you list more than one name, only one name needs to match for a match to occur. The community matching is effectively a logical OR operation. For more information, see "Configure Communities" on page 90.	
external [type *metric-type*]	Standard	(OSPF only) External routes, including routes exported from one level to another. type is an optional keyword. *metric* can either be 1 or 2. When you do not specify type, this condition matches all external routes. When you specify type, this condition matches only OSPF routes with the specified OSPF metric type.	
family *family-name*	Standard	Name of an address family. *family-name* can be either inet or inet6. Match the address family Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) of the route. Default setting is inet.	
instance *instance-name*	Standard	Routing instance or instances specified by name. Match a route learned from one of the specified instances.	Routing instance or instances specified by name. Match a route to be advertised over one of the specified instances.
interface *interface-name*	Standard	Router interface or interfaces specified by name or IP address. Do not use this qualifier with protocols that are not interface-specific, such as internal BGP (IBGP). Match a route learned from one of the specified interfaces. Direct routes match routes configured on the specified interface.	Router interface or interfaces specified by name or IP address. Do not use this Qualifier with protocols that are not interface specific, such as IBGP. Match a route to be advertised from one of the specified interfaces.
level *level*	Standard	(Intermediate System-to-Intermediate System [IS-IS] only) IS-IS level. Match a route learned from a specified level.	(IS-IS only) IS-IS level. Match a route to be advertised to a specified level.
local-preference *value*	Standard	(BGP only) BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range 0 through 4,294,967,295 (2^{32} – 1).	
metric *metric* metric2 *metric* metric3 *metric* metric4 *metric*	Standard	Metric value. You can specify up to four metric values, starting with metric (for the first metric value) and continuing with metric2, metric3, and metric4. (BGP only) metric corresponds to the multiple exit discriminator (MED), and metric2 corresponds to the interior gateway protocol (IGP) metric if the BGP next hop runs back through another route.	
neighbor *address*	Standard	Neighbor (peer) address or addresses. For BGP, the address can be a directly connected or indirectly connected peer. For all other protocols, the address is the neighbor from which the advertisement is received.	Neighbor (peer) address or addresses. For BGP import policies, specifying to neighbor produces the same result as specifying from neighbor. For BGP export policies, specifying the neighbor match condition has no effect and is ignored. For all other protocols, the to statement matches the neighbor to which the advertisement is sent.
next-hop *address*	Standard	Next-hop address or addresses specified in the routing information for a particular route. For BGP routes, matches are performed against the protocol next hop(s).	

Match Condition	Match Condition Category	from Statement Description	to Statement Description
origin *value*	Standard	(BGP only) BGP origin attribute, which is the origin of the AS path information. The value can be one of the following: ■ egp—Path information originated in another AS. ■ igp—Path information originated within the local AS. ■ incomplete—Path information was learned by some other means.	
policy [*policy-names*]	Extended	Name of a policy to evaluate as a subroutine. For information about this extended match condition, see "Configure Subroutines" on page 114.	
preference *preference* preference2 *preference*	Standard	Preference value. You can specify a primary preference value (preference) and a secondary preference value (preference2). The preference value can be a number from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. To specify even finer-grained preference values, see the color and color2 match conditions in this table. For more information about preference values, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.	
prefix-list *name* *ip-addresses*	Extended	Named list of IP addresses. You can specify an exact match with incoming routes. For information about this extended match condition, see "Configure Prefix Lists" on page 102.	You cannot specify this match condition.
protocol *protocol*	Standard	Name of the protocol from which the route was learned or to which the route is being advertised. It can be one of the following: aggregate, bgp, direct, dvmrp, isis, local, ospf, pim-dense, pim-sparse, rip, ripng, or static.	
rib *routing-table*	Standard	Name of a routing table. The value of *routing-table* can be one of the following: ■ inet.0—Unicast IPv4 routes ■ *instance-name*.inet.0—Unicast IPv4 routes for a particular routing instance ■ inet.1—Multicast IPv4 routes ■ inet.2—Unicast IPv4 routes for multicast reverse path forwarding (RPF) lookup ■ inet.3—Multiprotocol Label Switching (MPLS) routes ■ mpls.0—MPLS routes for label-switched path (LSP) next hops ■ inet6.0—Unicast IPv6 routes	
route-filter *destination-prefix* *match-type* <*actions*>	Extended	List of destination prefixes. When specifying a destination prefix, you can specify an exact match with a specific route or a less precise match using match types. You can configure either a common action that applies to the entire list or an action associated with each prefix. For more information, see "Configure Route Lists" on page 106.	You cannot specify this match condition.
source-address-filter *destination-prefix* *match-type* <*actions*>	Extended	List of multicast source addresses. When specifying a source address, you can specify an exact match with a specific route or a less precise match using match types. You can configure either a common action that applies to the entire list or an action associated with each prefix. For more information, see "Configure Route Lists" on page 106.	You cannot specify this match condition.
tag *string* tag2 *string*	Standard	You can specify two tag strings: tag (for the first string) and tag2. These values are local to the router and can be set on configured routes or by using an import routing policy. For OSPF only, the tag and tag2 match conditions match the 32-bit tag field in OSPF external link-state advertisement (LSA) packets.	

Actions

Each term can include a then statement, which defines the actions to take if a route matches all the conditions in the from and to statements:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                policy subroutine-policy-name;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
                policy subroutine-policy-name;
            }
            then {
                actions;
            }
        }
    }
}
```

If a term does not have from and to statements, all routes are considered to match and the actions apply to them all.

The then statement is optional.

You can specify one or more actions. There are three types of actions:

- Flow control actions, which affect whether to accept or reject the route and whether to evaluate the next term or routing policy.
- Actions that manipulate route characteristics.
- Trace action, which logs route matches.

When you specify an action that manipulates the route characteristics, the changes occur in a copy of the source route. The source route itself does not change. The effect of the action is visible only after the route is imported into or exported from the routing table. To view the source route before the routing policy has been applied, use the show route receive-protocol command. To view a route after an export policy has been applied, use the show route advertised-protocol command.

During policy evaluation, the characteristics in the copy of the source route always change immediately after the action is evaluated. However, the route is not copied to the routing table or a routing protocol until the completion of the policy evaluation is complete.

If you do not include a then statement, one of the following occurs:

- The next term in the routing policy, if one is present, is evaluated.
- If there are no more terms in the routing policy, the next routing policy, if one is present, is evaluated.
- If there are no more terms or routing policies, the accept or reject action specified by the default policy is taken. For more information, see "Default Routing Policies and Actions" on page 19.

Flow Control Actions

Table 10 lists the flow control actions. You can specify one of these actions along with the trace action (see "Trace Action" on page 48) or one or more of the actions that manipulate route characteristics (see "Actions That Manipulate Route Characteristics" on page 45).

Table 10: Flow Control Actions

Flow Control Action	Description
accept	Accept the route and propagate it. After a route is accepted, no other terms in the routing policy and no other routing policies are evaluated.
default-action accept	Accept and override any action intrinsic to the protocol. This is a nonterminating policy action.
reject	Reject the route and do not propagate it. After a route is rejected, no other terms in the routing policy and no other routing policies are evaluated.
default-action reject	Reject and override any action intrinsic to the protocol. This is a nonterminating policy action.
next term	Skip to and evaluate the next term in the same routing policy. Any accept or reject action specified in the then statement is skipped. Any actions in the then statement that manipulate route characteristics are applied to the route. next term is the default control action if a match occurs and you do not specify a flow control action.
next policy	Skip to and evaluate the next routing policy. Any accept or reject action specified in the then statement is skipped. Any actions in the then statement that manipulate route characteristics are applied to the route. next policy is the default control action if a match occurs, you do not specify a flow control action, and there are no further terms in the current routing policy.

Actions That Manipulate Route Characteristics

You can specify one or more of the actions listed in Table 11 to manipulate route characteristics.

Table 11: Actions That Manipulate Route Characteristics

Action	Description
as-path-prepend *as-path*	(BGP only) Affix one or more AS numbers at the beginning of the AS path. To specify more than one AS number, include the numbers in quotation marks. The AS numbers are added after the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed with a nonconfederation sequence. For more information, see "Configure AS Path Prepend Action" on page 119.
as-path-expand last-as count *n*	(BGP only) Extract the last AS number in the existing AS path and affix that AS number to the beginning of the AS path *n* times, where *n* is a number from 1 through 32. The AS number is added before the local AS number has been added to the path. This action adds AS numbers to AS sequences only. AS sets are ignored. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed with a nonconfederation sequence. This option is typically used in non-IBGP export policies.
class *class-name*	(Class of service only) Apply class-of-service parameters to routes installed into the routing table. For more information, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*.
color *preference* color2 *preference*	Set the preference value to a specific value. The color and color2 preference values are even finer grained than those specified in the preference and preference2 actions. The color value can be a number in the range 0 through 4,294,967,295 (2^{32} –1). A lower number indicates a more preferred route. If you set the preference with the color action, the value is internal to the JUNOS software and is not transitive. For more information about preference values, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.
color (add | subtract) *number* color2 (add | subtract) *number*	Change the color preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 (2^{32} –1), the value is set to 2^{32} –1. If a subtraction operation results in a value less than 0, the value is set to 0. If no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.
community (+ | add) [*names*]	(BGP only) Add communities to the set of communities in the route. For more information, see "Configure Communities" on page 90.
community (– | delete) [*names*]	(BGP only) Delete communities from the set of communities in the route. For more information, see "Configure Communities" on page 90.
community (= | set) [*names*]	(BGP only) Set the communities in the route, replacing any communities that were in the route. For more information, see "Configure Communities" on page 90.
damping *name*	(BGP only) Apply route-damping parameters to the route. These parameters override the default damping parameters. This action is useful only in an import policy, because the damping parameters affect the state of routes in the routing table. To apply damping parameters, you must enable BGP flap damping as described in the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*, and you must create a named list of parameters as described in "Configure Damping Action" on page 120.

Action	Description
destination-class *destination-class-name*	Maintain packet counts for a route passing through your network, based on the destination address in the packet. You can do the following: ■ Configure group destination prefixes by configuring a routing policy; see "Define Routing Policies" on page 37 and "Examples: Routing Policy Configuration" on page 62. ■ Apply that routing policy to the forwarding table with the corresponding destination class; see "Apply Routing Policies to the Forwarding Table" on page 60. For more information about the forwarding-table configuration statement, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*. ■ Enable packet counting on one or more interfaces by including the destination-class-usage statement at the [edit interfaces *interface-name* unit *logical-unit-number* family inet accounting] hierarchy level (see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*). See "Examples: Routing Policy Configuration" on page 62. ■ View the output by using one of the following commands: show interface destination-class *class-name interface-name*, show interface *interface-name* extensive, or show interface *interface-name* statistics (see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*). To configure a packet count based on the source address, use the source-class statement (SCU), described below.
external type *metric*	Set the external metric type for routes exported by OSPF. You must specify the keyword type.
forwarding-class *forwarding-class-name*	Create the forwarding-class which includes packets based on both the destination address and the source address in the packet. You can do the following: ■ Configure group prefixes by configuring a routing policy; see "Define Routing Policies" on page 37 and "Examples: Routing Policy Configuration" on page 62. ■ Apply that routing policy to the forwarding table with the corresponding forwarding class; see "Apply Routing Policies to the Forwarding Table" on page 60. For more information about the forwarding-table configuration statement, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*. ■ Enable packet counting on one or more interfaces by using the procedure described in either the destination-class or source-class actions, which are defined elsewhere in this table.
install-nexthop lsp *lsp-name*	Choose which next hops, among a set of equal LSP next hops, are installed in the forwarding table. Use the export policy for the forwarding table to specify the LSP next hop to be used for the desired routes.
load balance per-packet	(For export to the forwarding table only) Install all next-hop addresses into the forwarding table and have the forwarding table perform per-packet load balancing. For more information, see "Configure Load-Balance Per-Packet Action" on page 125.
local-preference *value*	(BGP only) Set the BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range from 0 through 4,294,967,295.
local-preference (add | subtract) *number*	Change the local preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value. For BGP, if the attribute value is not known, the local preference attribute value is initialized to 100 before the routing policy is applied.
metric *metric* metric2 *metric* metric3 *metric* metric4 *metric*	Set the metric. You can specify up to four metric values, starting with metric (for the first metric value) and continuing with metric2, metric3, and metric4. (BGP only) metric corresponds to the MED, and metric2 corresponds to the IGP metric if the BGP next hop loops through another router.
metric (add | subtract) *number* metric2 (add | subtract) *number* metric3 (add | subtract) *number* metric4 (add | subtract) *number*	Change the metric value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.
metric (igp | minimum-igp) *site-offset*	(BGP only) Change the metric (MED) value by the specified negative or positive offset. This action is useful only in an external BGP (EBGP) export policy.
next-hop (*address* | peer-address)	Set the next hop. When the advertising protocol is BGP, you can set the next hop only when any third-party next hop can be advertised; that is, when using IBGP or EBGP confederations. If you specify *address* as self, the next-hop address is replaced by one of the local router's addresses. The advertising protocol determines which address to use. When the advertising protocol is BGP, this address is set to the local IP address used for the BGP adjacency. A router cannot install routes with itself as the next hop. If you specify peer-address, the next-hop address is replaced by the peer's IP address. This option is valid only in import policies. Primarily used by BGP to enforce using the peer's IP address for advertised routes, this option is meaningful only when the next hop is the advertising router or another directly connected router.

Action	Description
origin *value*	(BGP only) Set the BGP origin attribute to one of the following values: ■ igp—Path information originated within the local AS. ■ egp—Path information originated in another AS. ■ incomplete—Path information was learned by some other means.
preference *preference* preference2 *preference*	Set the preference value. You can specify a primary preference value (preference) and a secondary preference value (preference2). The preference value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. To specify even finer-grained preference values, see the color and color2 actions in this table. If you set the preference with the preference action, the new preference remains associated with the route. The new preference is internal to the JUNOS software and is not transitive. For more information about preference values, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.
preference (add | subtract) *number* preference2 (add | subtract) *number*	Change the preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.
source-class *source-class-name*	Maintain packet counts for a route passing through your network, based on the source address. You can do the following: ■ Configure group source prefixes by configuring a routing policy; see "Define Routing Policies" on page 37 and "Examples: Routing Policy Configuration" on page 62. ■ Apply that routing policy to the forwarding table with the corresponding source class; see "Apply Routing Policies to the Forwarding Table" on page 60. For more information about the forwarding-table configuration statement, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*. ■ Enable packet counting on one or more interfaces by including the source-class-usage statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name accounting*] hierarchy level (see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*). Also, follow the source-class-usage statement with the input or output statement to define the inbound and outbound interfaces on which SCU-monitored traffic is arriving and departing (or define one interface for both). The complete syntax is [edit interfaces *interface-name* unit *unit-number* family inet accounting source-class-usage (input | output | [input output]). See the example in "Examples: Routing Policy Configuration" on page 62. ■ View the output by using one of the following commands: show interface source-class *class-name interface-name*, show interface *interface-name* extensive, or show interface *interface-name* statistics (see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*). To configure a packet count based on the destination address, use the destination-class statement (DCU), described above.
tag *tag* tag2 *tag*	You can specify two tag strings: tag (for the first string) and tag2. These values are local to the router. (For OSPF and IS-IS only) The tag and tag2 actions set the 32-bit tag field in OSPF external LSA packets, and the 32-bit flag in the IS-IS IP prefix type length values (TLV).
tag (add | subtract) *number* tag2 (add | subtract) *number*	Change the tag value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.

Trace Action

If you specify the trace action, the match is logged to a trace file. To set up a trace file, you must specify the following elements in the global traceoptions statement:

- Trace filename
- policy option in the flag statement

For more information about the global traceoptions statement, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

The following example uses the trace filename of policy-log:

```
[edit]
routing-options {
    traceoptions {
        file “policy-log”;
        flag policy;
    }
}
```

This action does not affect the flow control during routing policy evaluation.

If a term that specifies a trace action also specifies a flow control action, the name of the term will be logged in the trace file. If a term specifies a trace action only, the word <default> only will be logged.

Final Action

In addition to specifying an action using the then statement in a named term, you can also specify an action using the then statement in an unnamed term, as follows:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                policy subroutine-policy-name;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
                policy subroutine-policy-name;
            }
            then {
                actions;
            }
        }
        then action;
    }
}
```

Default Action

This default-action statement overrides any action intrinsic to the protocol. This action is also nonterminating, so that various policy terms can be evaluated before the policy is terminated. You can specify a default action, either accept or reject, as follows:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                family family-name
                match-conditions;
                policy subroutine-policy-name;
                prefix-list name;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            to {
                match-conditions;
                policy subroutine-policy-name;
            }
            then {
                actions;
                default-action (accept | reject);
            }
        }
    }
}
```

The resulting action is set either by the protocol or by the last policy term that is met.

Example: Configure the Default Action for a Policy

Configure a routing policy that matches routes based on three policy terms. If the route matches the first term, a certain community tag is attached. If the route matches two separate terms, then both community tags are attached. If the route does not match any terms, it is rejected (protocol's default action). Note that the terms hub and spoke are mutually exclusive.

```
[edit]
policy-options {
    policy-statement test {
        term set-default {
            then default-action reject;
        }
        term hub {
            from interface ge-2/1/0.5;
            then {
                community add test-01-hub;
                default-action accept;
            }
        }
        term spoke {
            from interface [ ge-2/1/0.1 ge-2/1/0.2 ];
            then {
                community add test-01-spoke;
                default-action accept;
            }
        }
```

```
        term management {
            from protocol direct;
            then {
                community add management;
                default-action accept;
            }
        }
    }
}
```

Route List Actions

If you specify route lists in the from statement, for each route in the list, you can specify an action to take on that individual route directly, without including a then statement. For more information, see “Configure Route Lists” on page 106.

Examples: Define Routing Policies

This section provides examples of defining routing policies. For more examples, see “Examples: Routing Policy Configuration” on page 62.

Define a Routing Policy from BGP to IS-IS

Accept BGP routes advertised by the peer 128.125.1.1. If a route matches, it is accepted, and no further evaluation is performed on that route. If a route does not match, the accept or reject action specified by the default policy is taken. (For more information about the default routing policies, see “Default Routing Policies and Actions” on page 19.) If you apply this routing policy to imported BGP routes, only the routes learned from the peer 128.125.1.1 and BGP transit routes are accepted from BGP peers.

```
[edit]
policy-options {
    policy-statement bgp-to-isis {
        term term1 {
            from {
                neighbor 128.125.1.1;
            }
            then {
                accept;
            }
        }
    }
}
```

Use Routing Policy to Set Preference

Define a routing policy that matches routes from specific next hops that are being advertised to specific neighbors and that sets the preference. If a route does not match the first term, it is evaluated by the second term. If it still does not match, the next routing policy, if configured, is evaluated; then the accept or reject action specified by the default policy is taken. (For more information about the default routing policies, see “Default Routing Policies and Actions” on page 19.)

```
[edit]
policy-options {
    policy-statement set-preference {
        term term1 {
            from {
                next-hop [10.0.0.1 10.0.0.2];
            }
            to {
                neighbor 128.1.1.1;
            }
            then {
                preference 10;
            }
        }
        term term2 {
            from {
                next-hop 10.0.0.3;
            }
            to {
                neighbor 128.2.1.1;
            }
            then {
                preference 15;
            }
        }
    }
}
```

Apply Routing Policies

For a routing policy to take effect, you must apply it to either a routing protocol or the forwarding table. This section contains the following information:

- Apply Routing Policies to a Routing Protocol on page 52
- Apply Routing Policies to the Forwarding Table on page 60
- Examples: Apply Routing Policies on page 61

Apply Routing Policies to a Routing Protocol

Before applying routing policies to routing protocols, you must know if each protocol supports import and export policies and the level at which you can apply these policies. Table 8 on page 26 summarizes the import and export policy support for each routing protocol and the level at which you can apply these policies.

For more information about applying routing policies to individual routing protocols, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

This section describes the following tasks for applying routing policies to a routing protocol and some side effects of these tasks:

Apply a Routing Policy

To apply a routing policy to a routing protocol, include the import and export statements at the [edit protocols *protocol-name*] hierarchy level:

```
[edit protocols protocol-name]
import [ policy-names ];
export [ policy-names ];
```

In the import statement, list the name of the routing policy to be evaluated when routes are imported into the routing table from the routing protocol.

In the export statement, list the name of the routing policy to be evaluated when routes are being exported from the routing table into a dynamic routing protocol. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or different import and export statements.

For information about how the policy framework software evaluates a routing policy, see “How a Routing Policy Is Evaluated” on page 27.

Apply a Routing Policy Chain

To apply multiple routing policies (chains) to a routing protocol, include the import and export statements at the [edit protocols *protocol-name*] hierarchy level:

```
[edit protocols protocol-name]
import [ policy-names ];
export [ policy-names ];
```

In the import statement, list the names of multiple routing policies to be evaluated when routes are imported into the routing table from the routing protocol.

In the export statement, list the names of multiple routing policies to be evaluated when routes are being exported from the routing table into a dynamic routing protocol. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or different import and export statements.

The policy framework software evaluates the routing policies sequentially, from left to right. If an action specified in one of the policies manipulates a route characteristic, the policy framework software carries the new route characteristic forward during the evaluation of the remaining policies. For example, if the action specified in the first policy of a chain sets a route's metric to 500, this route matches the criterion of metric 500 defined in the next policy.

For more information about routing policy chain evaluation, see "How a Routing Policy Chain Is Evaluated" on page 28.

Apply Policy Expressions

Policy expressions give the policy framework software a different way to evaluate routing policies. A *policy expression* uses Boolean logical operators with policies. The logical operators establish rules by which the policies are evaluated.

During evaluation of a routing policy in a policy expression, the policy action of accept, reject, or next policy is converted to the value of TRUE or FALSE. This value is then evaluated against the specified logical operator to produce output of either TRUE or FALSE. The output is then converted back to a flow control action of accept, reject, or next policy. The result of the policy expression is applied as it would be applied to a single policy; the route is accepted or rejected and the evaluation ends, or the next policy is evaluated.

Table 12 summarizes the policy actions and their corresponding TRUE and FALSE values and flow control action values. Table 13 describes the logical operators. For complete information about policy expression evaluation, see "How a Policy Expression Is Evaluated" on page 56.

You must enclose a policy expression in parentheses. You can place a policy expression anywhere in the import or export statements and in the from policy statement.

Table 12: Policy Action Conversion Values

Policy Action	Conversion Value	Flow Control Action Conversion Value
Accept	TRUE	Accept
Reject	FALSE	Reject
Next policy	TRUE	Next policy

Table 13: Policy Expression Logical Operators

Logical Operator	Policy Expression Logic	How Logical Operator Affects Policy Expression Evaluation
&& (Logical AND)	Logical AND requires that all values must be TRUE to produce output of TRUE. Routing policy value of TRUE and TRUE produces output of TRUE. Value of TRUE and FALSE produces output of FALSE. Value of FALSE and FALSE produces output of FALSE.	If the first routing policy returns the value of TRUE, the next policy is evaluated. If the first policy returns the value of FALSE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated.
|| (Logical OR)	Logical OR requires that at least one value must be TRUE to produce output of TRUE. Routing policy value of TRUE and FALSE produces output of TRUE. Value of TRUE and TRUE produces output of TRUE. Value of FALSE and FALSE produces output of FALSE.	If the first routing policy returns the value of TRUE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If the first policy returns the value of FALSE, the next policy is evaluated.
! (Logical NOT)	Logical NOT reverses value of TRUE to FALSE and of FALSE to TRUE. It also reverses the actions of accept and next policy to reject, and reject to accept.	If used with the logical AND operator and the first routing policy value of FALSE is reversed to TRUE, the next policy is evaluated. If the value of TRUE is reversed to FALSE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If used with the logical OR operator and the first routing policy value of FALSE is reversed to TRUE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If the value of TRUE is reversed to FALSE, the next policy is evaluated. If used with a policy and the flow control action is accept or next policy, these actions are reversed to reject. If the flow control action is reject, this action is reversed to accept.

Policy Expression Examples

The following examples show how to use the logical operators to create policy expressions:

- Logical AND—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, policy2 is evaluated. If a value of FALSE is returned, policy2 is not evaluated.

```
export (policy1 && policy2)
```

- Logical OR—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, policy2 is not evaluated. If a value of FALSE is returned, policy2 is evaluated.

```
export (policy1 || policy2)
```

- Logical OR and logical AND—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, policy2 is skipped and policy3 is evaluated. If after evaluating policy1, a value of FALSE is returned, policy2 is evaluated. If policy2 returns a value of TRUE, policy3 is evaluated. If policy2 returns a value of FALSE, policy3 is not evaluated.

  ```
  export [(policy1 || policy2) && policy3]
  ```

- Logical NOT—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, the value is reversed to FALSE and policy2 is not evaluated. If a value of FALSE is returned, the value is reversed to TRUE and policy2 is evaluated.

  ```
  export (!policy1 && policy2)
  ```

The sequential list [policy1 policy2 policy3] is not the same as the policy expression (policy1 && policy2 && policy3).

The sequential list is evaluated on the basis of a route matching a routing policy. For example, if policy1 matches and the action is accept or reject, policy2 and policy3 are not evaluated. If policy1 does not match, policy2 is evaluated and so on until a match occurs and the action is accept or reject.

The policy expressions are evaluated on the basis of the action in a routing policy that is converted to the value of TRUE or FALSE and the logic of the specified logical operator. (For complete information about policy expression evaluation, see “How a Policy Expression Is Evaluated” on page 56.) For example, if policy1 returns a value of FALSE, policy2 and policy3 are not evaluated. If policy1 returns a value of TRUE, policy2 is evaluated. If policy2 returns a value of FALSE, policy3 is not evaluated. If policy2 returns a value of TRUE, policy3 is evaluated.

You can also combine policy expressions and sequential lists. In the following example, if policy1 returns a value of FALSE, policy2 is evaluated. If policy2 returns a value of TRUE and contains a next policy action, policy3 is evaluated. If policy2 returns a value of TRUE but does not contain an action, including a next policy action, policy3 is still evaluated (because if you do not specify an action, next term or next policy are the default actions). If policy2 returns a value of TRUE and contains an accept action, policy3 is not evaluated.

```
export [(policy1 || policy2) policy3]
```

How a Policy Expression Is Evaluated

During evaluation, the policy framework software converts policy actions to values of TRUE or FALSE, which are factors in determining the flow control action that is performed upon a route. However, the software does not actually perform a flow control action on a route until it evaluates an entire policy expression.

The policy framework software evaluates a policy expression as follows:

1. The software evaluates a route against the first routing policy in a policy expression and converts the specified or default action to a value of TRUE or FALSE. (For information about the policy action conversion values, see Table 12 on page 54.)

2. The software takes the value of TRUE or FALSE and evaluates it against the logical operator used in the policy expression (see Table 13 on page 54). Based upon the logical operator used, the software determines whether or not to evaluate the next routing policy, if one is present.

   The policy framework software uses a method of shortcut evaluation. When a result is certain, the software stops evaluating subsequent routing policies in the policy expression. For example, if the policy expression specifies logical AND and the evaluation of the first routing policy returns the value of FALSE, the software determines that the output will be FALSE no matter what the values of the unevaluated routing policies are. Therefore, the software does not evaluate the subsequent routing policies in this policy expression.

3. The software performs Steps 1and 2 for each subsequent routing policy in the policy expression, if they are present and if the software has determined that it is appropriate to evaluate them.

4. After evaluating the last routing policy, if it is appropriate, the software evaluates the value of TRUE or FALSE obtained from each routing policy evaluation. Based upon the logical operator used, it calculates an output of TRUE or FALSE.

5. The software converts the output of TRUE or FALSE back to an action. (For information about the policy action conversion values, see Table 12 on page 54.) The action is performed.

   If each policy in the expression returned a value of TRUE, the software converts the output of TRUE back to the flow control action specified in the last policy. For example, if the policy expression (policy1 && policy2) is specified and policy1 specifies accept and policy2 specifies next term, the next term action is performed.

If an action specified in one of the policies manipulates a route characteristic, the policy framework software carries the new route characteristic forward during the evaluation of the remaining policies. For example, if the action specified in the first policy of a policy expression sets a route's metric to 500, this route matches the criteria of metric 500 defined in the next policy. However, if a route characteristic manipulation action is specified in a policy located in the middle or the end of a policy expression, it is possible, because of the shortcut evaluation, that the policy is never evaluated and the manipulation of the route characteristic never occurs.

Policy Expression Evaluation Example

The following sample routing policy uses three policy expressions:

```
[edit policy-options]
policy-options {
    policy-statement policy-A {
        from {
            route-filter 10.10.0.0/16 orlonger;
        }
        then reject;
    }
}

policy-options {
    policy-statement policy-B {
        from {
            route-filter 10.20.0.0/16 orlonger;
        }
        then accept;
    }
}

protocols {
    bgp {
        neighbor 1.1.1.1;
            export (policy-A && policy-B);
        }
        neighbor 1.1.2.1;
            export (policy-A || policy-B);
        }
        neighbor 1.1.3.1;
            export (!policy-A);
        }
    }
}
```

The policy framework software evaluates the transit BGP route 10.10.1.0/24 against the three policy expressions specified in the sample routing policy as follows:

- (policy-A && policy-B)—10.10.1.0/24 is evaluated against policy-A. 10.10.1.0/24 matches the route list specified in policy-A, so the specified action of reject is returned. reject is converted to a value of FALSE, then FALSE is evaluated against the specified logical AND. Because the result of FALSE is certain no matter what the results of the evaluation of policy-B are (in policy expression logic, any result AND a value of FALSE produces the output of FALSE), policy-B is not evaluated and the output of FALSE is produced. The FALSE output is converted to "reject," and 10.10.1.0/24 is rejected.

- (policy-A || policy-B)—10.10.1.0/24 is evaluated against policy-A. 10.10.1.0/24 matches the route list specified in policy-A, so the specified action of reject is returned. reject is converted to a value of FALSE, then FALSE is evaluated against the specified logical OR. Because logical OR requires at least one value of TRUE to produce an output of TRUE, 10.10.1.0/24 is evaluated against policy-B. 10.10.1.0/24 does not match policy-B, so the default action of "next policy" is returned. "Next policy" is converted to a value of TRUE, then the value of FALSE (for policy-A evaluation) and TRUE (for policy-B

evaluation) are evaluated against the specified logical OR. In policy expression logic, FALSE OR TRUE produce an output of TRUE. The output of TRUE is converted to "next policy." (TRUE is converted to "next policy" because "next policy" was the last action retained by the policy framework software.) policy-B is the last routing policy in the policy expression, so the action specified by the default export policy for BGP, "accept," is taken.

- (!policy-A)—10.10.1.0/24 is evaluated against policy-A. 10.10.1.0/24 matches the route list specified in policy-A, so the specified action of reject is returned. reject is converted to a value of FALSE, then FALSE is evaluated against the specified logical NOT. The value of FALSE is reversed to an output of TRUE based on the rules of logical NOT. The output of TRUE is converted to "accept," and route 10.10.1.0/24 is accepted.

Side Effects of Omitting the "from" Statement from an Export Policy

In export policies, omitting the from statement in a term might lead to unexpected results. By default, if you omit the from statement, all routes are considered to match. For example, static and direct routes are not exported by BGP by default. However, if you create a term with an empty from statement, these routes inadvertently could be exported because they matched the from statement. For example, the following routing policy is designed to reject a few route ranges and then export routes learned by BGP (which is the default export behavior):

```
[edit]
routing-options {
    autonomous-system 56;
}
protocols {
    bgp {
        group 4 {
            export statics-policy;
            type external;
            peer-as 47;
            neighbor 1.2.2.4;
        }
    }
}
policy-options {
    policy-statement statics-policy {
        term term1 {
            from {
                route-filter 192.168.0.0/16 orlonger;
                route-filter 224.0.0.0/3 orlonger;
            }
            then reject;                    # reject the prefixes in the route list
        }
        term term2 {
            then {
                accept;                     # accept all other routes, including static and direct routes
            }
        }
    }
}
```

However, this routing policy results in BGP advertising static and direct routes to its peers because:

- term1 rejects the destination prefixes enumerated in the route list.
- term2, because it has no from statement, matches all other routes, including static and direct routes, and accepts all these routes (with the accept statement).

To modify the routing policy shown above so that an IGP does not export unwanted routes, you can specify the following additional terms:

```
[edit]
routing-options {
    autonomous-system 56;
}
protocols {
    isis {
        export statics-policy;
    }
}
policy-options {
    policy-statement statics-policy {
        term term1 {
            from {
                route-filter 192.168.0.0/16 orlonger;
                route-filter 224.0.0.0/3 orlonger;
            }
            then reject;                    # reject the prefixes in the route list
        }
        term term2 {                        # reject direct routes
            from protocol direct;
            then reject;
        }
        term term3 {                        # reject static routes
            from protocol static;
            then reject;
        }
        term term4 {                        # reject local routes
            from protocol local;
            then reject;
        }
        term term5 {                        # reject aggregate routes
            from protocol aggregate;
            then reject;
        }
        term term6 {
            then accept;                    # accept all other routes
        }
    }
}
```

Apply Routing Policies to the Forwarding Table

To apply an export routing policy to the forwarding table, include the export statement at the [edit routing-options forwarding-table] hierarchy level:

```
[edit]
routing-options {
    forwarding-table {
        export [ policy-names ];
    }
}
```

In the export statement, list the name of the routing policy to be evaluated when routes are being exported from the routing table into the forwarding table. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or a different export statement.

For information about how the policy framework software evaluates a routing policy, see "How a Routing Policy Is Evaluated" on page 27.

You can apply export policies to routes being exported from the routing table into the forwarding table for the following features:

- Per-packet load balancing
- Class of service (CoS)

For more information about per-packet load balancing, see "Configure Load-Balance Per-Packet Action" on page 125. For more information about CoS, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

Examples: Apply Routing Policies

Configure the router to export to IS-IS the routes that match the dmz and local-customers routing policies:

```
[edit]
protocols {
    isis {
        export [dmz local-customers];
    }
}
```

For three BGP peer groups, apply various export and import filters:

```
[edit]
protocols {
    bgp {
        group 1 {
            type external;
            peer-as 47;
            export local-customers;
            import [martian-filter long-prefix-filter as47-filter];
            neighbor 1.2.2.4;
            neighbor 1.2.2.5;
        }
        group 2 {
            type external;
            peer-as 42;
            export local-customers;
            import [martian-filter long-prefix-filter as42-filter];
            neighbor 2.1.2.4;
            neighbor 2.1.2.5;
        }
        group 3 {
            type internal;
            export local-customers;
            neighbor 10.1.1.1;
        }
    }
}
```

Apply the long-prefix-filter prefix only to routes learned from a particular peer within a group:

```
[edit]
protocols {
    bgp {
        group 4 {
            type external;
            peer-as 47;
            export local-customers;
            import [martian-filter as47-filter];
            neighbor 1.2.2.4;
            neighbor 1.2.2.5;
            neighbor 1.2.2.6 {
                import [martian-filter as47-filter long-prefix-filter];
            }
        }
    }
}
```

Examples: Routing Policy Configuration

Redistribute BGP routes with a community tag of 444:5 into IS-IS, changing the metric to 14:

```
[edit]
protocols {
    isis {
        export edu-to-isis;
    }
}
policy-options {
    community edu members 444:5;
    policy-statement edu-to-isis {
        from {
            protocol bgp;
            community edu;
        }
        then {
            metric 14;
            accept;
        }
    }
}
```

Redistribute OSPF routes from area 1 only into BGP, and do not advertise routes learned by BGP:

```
[edit]
routing-options {
    autonomous-system 56;
}
protocols {
    bgp {
        export ospf-into-bgp;
        group {
            type external;
            peer-as 23;
            allow {
                0.0.0.0/0;
            }
        }
    }
}
policy-options {
    policy-statement ospf-into-bgp {
        term ospf-only {
            from {
                protocol ospf;
                area 1;
            then accept;
            }
        }
    }
}
```

Define a routing policy to export direct routes into IS-IS for all interfaces, even if IS-IS is not configured on an interface:

```
[edit]
protocols {
    isis {
        export direct-routes;
    }
}
policy-options {
    policy-statement direct-routes {
        from protocol direct;
        then accept;
    }
}
```

Define a routing policy to export IS-IS Level 1 internal-only routes into Level 2:

```
[edit]
protocols {
    isis {
        export L1-L2;
    }
}
policy-statement L1-L2 {
    term one {
        from {
            level 1;
            external;
        }
        then reject;
    }
    term two {
        from level 1;
        to level 2;
        then accept;
    }
}
```

Define a routing policy to export IS-IS Level 2 routes into Level 1:

```
[edit]
protocols {
    isis {
        export L2-L1;
    }
}
policy-statement L2-L1 {
    term one {
        from level 2;
        to level 1;
        then accept;
    }
}
```

Configure different forwarding next-hop LSPs for different destination prefixes learned from BGP:

```
routing-options {
    router-id 10.10.20.101;
    autonomous-system 2;
    forwarding-table {
        export forwarding-policy;
    }
}
policy-options {
    policy-statement forwarding-policy {
        term one {
            from {
                protocol bgp;
                route-filter 10.1.0.0/16 orlonger;
            }
            then {
                install-nexthop lsp mc-c-lsp-1;
            }
        }
        term two {
            from {
                protocol bgp;
                route-filter 10.2.0.0/16 orlonger;
                }
            then {
                install-nexthop lsp mc-c-lsp-2;
            }
        }
        term three {
            from {
                protocol bgp;
                route-filter 10.3.0.0/16 orlonger;
            }
            then {
                install-nexthop lsp mc-c-lsp-3;
            }
        }
    }
}
protocols {
    mpls {
        label-switched-path mc-c-lsp-1 {
            from 10.10.20.101;
            to 10.10.20.103;
        }
        label-switched-path mc-c-lsp-2 {
            from 10.10.20.101;
            to 10.10.20.103;
        }
        label-switched-path mc-c-lsp-3 {
            from 10.10.20.101;
            to 10.10.20.103;
        }
    }
}
```

Configure a routing policy to group destination prefixes:

```
[edit]
policy-options {
   policy-statement set-dest-class {
      term 1 {
         from community nets1;
         then {
            destination-class on-net;
            accept;
         }
      }
      term 2 {
         from community nets2;
         then {
            destination-class off-net;
            accept;
         }
      }
   }
   community nets1 [7:8 9:10];
   community nets2 [1:2 4:5];
```

Apply a routing policy to the forwarding table with the corresponding destination class:

```
[edit]
routing-options {
   forwarding-table {
      export set-dest-class;
   }
}
```

Enable packet counting on an interface:

```
[edit interfaces]
interfaces so-1/0/1 {
   unit 0 {
      family inet6 {
         accounting {
          destination-class-usage;
         }
      }
   }
}
```

Configure a routing policy to group source prefixes, and allow prefixes that match the policy statement to have a source class created for them:

```
[edit]
policy-options {
   policy-statement set-gold-class {
      term {
         from
            route-filter 210.210.0.0/16 orlonger;
            route-filter 215.215.0.0/16 orlonger;
         then {
            source-class gold-class;
```

Apply a routing policy to the forwarding table with the corresponding source class:

```
[edit]
routing-options {
    forwarding-table {
        export set-gold-class;
    }
}
```

Enable packet counting on an interface. In this example, one interface accommodates both input and output:

```
[edit interfaces]
interfaces ge/0/0/0 {
    unit 0 {
        family inet {
            accounting {
                source-class-usage {
                    input;
                    output;
                }
            }
        }
    }
}
```

Configure a routing policy to group source and destination prefixes in a forwarding class:

```
[edit]
policy-options {
    policy-statement set-bronze-class {
        term {
            from
                route-filter 210.210.0.0/16 orlonger;
                route-filter 215.215.0.0/16 orlonger;
            then {
                forwarding-class bronze-class;
```

Apply a routing policy to the forwarding table with the corresponding forwarding class:

```
[edit]
routing-options {
    forwarding-table {
        export set-bronze-class;
    }
}
```

Enable counting of incoming source packets on an interface:

```
[edit interfaces]
interfaces fe/1/0/0 {
    unit 0 {
        family inet {
            accounting {
                source-class-usage {
                    input;
                }
            }
        }
    }
}
interfaces fe/1/0/1 {
    unit 0 {
        family inet {
            accounting {
                source-class-usage {
                    output;
                }
            }
        }
    }
}
interfaces fe/1/0/2 {
    unit 0 {
        family inet {
            accounting {
                destination-class-usage;
            }
        }
    }
}
```

Configure a policy that accepts routes with the destination prefixes fe80::90:69ff:fea0:8000/128 and fe80::90:69ff:fea0:8001/128:

```
[edit policy-options]
policy-statement export_exact {
term a {
    from {
        route-filter fe80::90:69ff:fea0:8000/128 exact;
        route-filter fe80::90:69ff:fea0:8001/128 exact;
        }
    then {
        accept;
        }
    }
term b {
    then {
        reject;
        }
    }
}
```

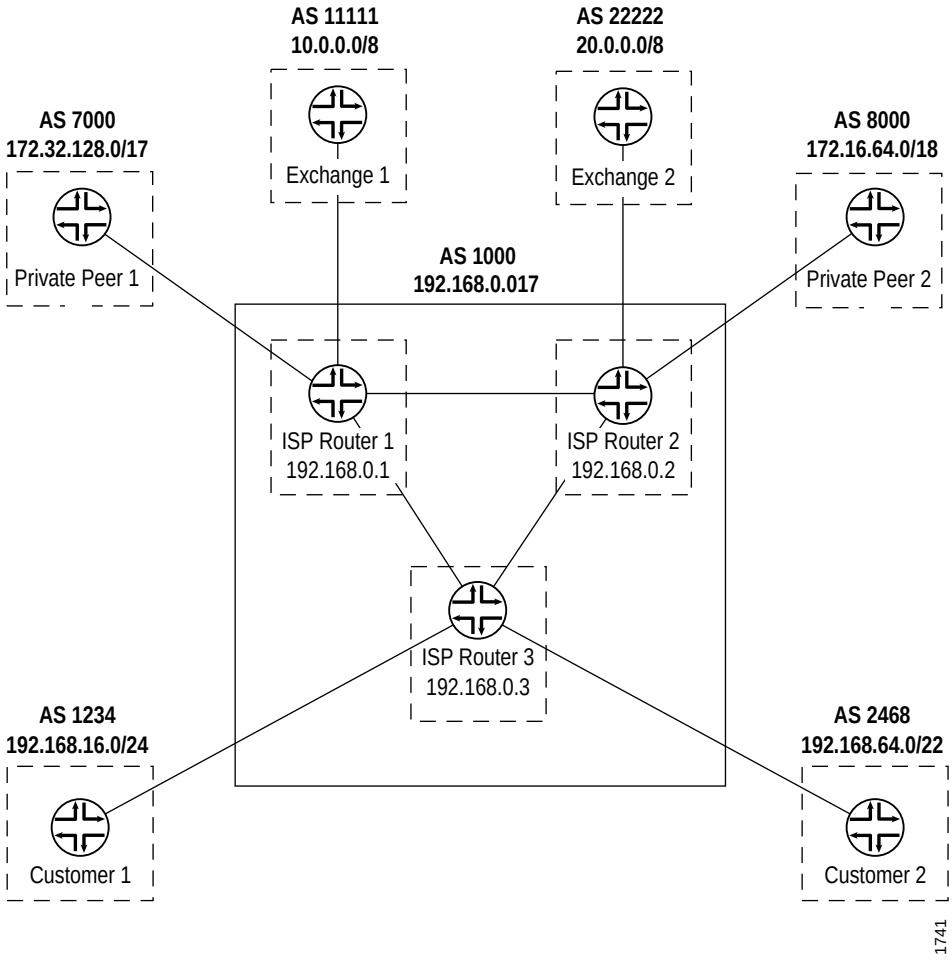
Configure a policy that accepts routes with the destination prefix fec0:1001:2:3::/64:

```
[edit policy-options]
policy-statement export_exact {
term a {
    from {
        protocol bgp;
        route-filter fec0:1001:2:3::/64 exact;
        }
    then {
        accept;
        }
    }
term b {
    then {
        reject;
        }
    }
}
```

Example: ISP Network Case Study

This section provides an example of how policies might be used in a typical Internet service provider (ISP) network. In this network example (see Figure 11), the ISP's AS number is 1000. The ISP has two transit peers (AS 11111 and AS 22222) to which it connects at an exchange point. The ISP is also connected to two private peers (AS 7000 and AS 8000) with which it exchanges specific customer routes. The ISP has two customers (AS 1234 and AS 2468) to which it connects to using the BGP protocol.

Figure 11: ISP Network Example



In this example, the ISP policies are configured in an outbound direction; that is, the example focuses on the routes that the ISP announces to its peers and customers, and includes the following:

1. The ISP has been assigned AS 1000 and the routing space of 192.168.0/17. With the exception of the two customer networks shown in Figure 11, all other customer routes are simulated with static routes.

2. The ISP has connectivity to two different exchange peers: AS 11111 and AS 22222. These peers are used for transit service to other portions of the Internet. This means that the ISP is accepting all routes (the full Internet routing table) from those BGP peers. To help maintain an optimized Internet routing table, the ISP is configured to advertise only two aggregate routes to the transit peers.

3. The ISP also has direct connectivity to two private peers: AS 7000 and AS 8000. The ISP administrators want all data to the private peers to use this direct link. As a result, all the customer routes from the ISP are advertised to those private peers. These peers then advertise all their customer routes to the ISP.

4. Finally, the ISP has two customers with which it communicates using BGP: AS 1234 and AS 2468. Each customer has a different set of requirements.

Request a Single Default Route on the Customer 1 Router

Customer 1 has only a single route to the ISP and is using the ISP for transit service. This customer has requested a single default route (0.0.0.0/0) from the ISP.

```
[edit]
interfaces {
    so-0/0/1 {
        description "Connection to ISP Router 3";
        unit 0 {
            family inet {
                address 10.222.70.1/30;
            }
        }
    }
    fxp0 {
        description "MGMT INTERFACE - DO NOT DELETE";
        unit 0 {
            family inet {
                address 10.251.0.9/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 182.168.16.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 192.168.16.0/27 reject;
        route 192.168.16.32/27 reject;
        route 192.168.16.64/27 reject;
        route 192.168.16.96/27 reject;
        route 192.168.16.128/27 reject;
        route 192.168.16.160/27 reject;
        route 192.168.16.192/27 reject;
    }
    autonomous-system 1234;
}
```

```
protocols {
    bgp {
        group AS1000-Peers {
            type external;
            export send-statics;
            peer-as 1000;
            neighbor 10.222.70.2;
        }
    }
}
policy-options {
    policy-statement send-statics {
        term static-routes {
            from protocol static;
            then accept;
        }
    }
}
```

Request Specific Routes on the Customer 2 Router

Customer 2 has a link to the ISP, as well as a link to AS 8000. This customer has requested specific customer routes from the ISP, as well as from AS 8000. Customer 2 wants to use the ISP for transit service to the Internet, and has requested a default route from the ISP.

```
[edit]
interfaces {
    so-0/0/1 {
        description "Connection to ISP Router 3";
        unit 0 {
            family inet {
                address 10.222.61.2/30;
            }
        }
    }
    so-0/0/2 {
        description "Connection to Private-Peer 2";
        unit 0 {
            family inet {
                address 10.222.6.1/30;
            }
        }
    }
    fxp0 {
        description "MGMT INTERFACE - DO NOT DELETE";
        unit 0 {
            family inet {
                address 10.251.0.8/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.64.1/32;
            }
        }
    }
}
```

```
routing-options {
    static {
        route 192.168.64.0/25 reject;
        route 192.168.64.128/25 reject;
        route 192.168.65.0/25 reject;
        route 192.168.66.0/25 reject;
        route 192.168.67.0/25 reject;
        route 192.168.65.128/25 reject;
        route 192.168.66.128/25 reject;
        route 192.168.67.128/25 reject;
    }
    autonomous-system 2468;
}
protocols {
    bgp {
        group External-Peers {
            type external;
            import inbound-routes;
            export outbound-routes;
            neighbor 10.222.61.1 {
                peer-as 1000;
            }
            neighbor 10.222.6.2 {
                peer-as 8000;
            }
        }
    }
}
policy-options {
    policy-statement outbound-routes {
        term statics {
            from protocol static;
            then accept;
        }
        term internal-bgp-routes {
            from {
                protocol bgp;
                as-path my-own-routes;
            }
            then accept;
        }
        term no-transit {
            then reject;
        }
    }
    policy-statement inbound-routes {
        term AS1000-primary {
            from {
                protocol bgp;
                as-path AS1000-routes;
            }
            then {
                local-preference 200;
                accept;
            }
        }
```

```
        term AS8000-backup {
            from {
                protocol bgp;
                as-path AS8000-routes;
            }
            then {
                local-preference 50;
                accept;
            }
        }
    }
    as-path my-own-routes "()";
    as-path AS1000-routes "1000 .*";
    as-path AS8000-routes "8000 .*";
}
```

Configure Peer Policy on ISP Router 3

On ISP Router 3, a separate policy is in place for each customer. The default route for Customer 1 is being sent by the customer-1-peer policy. This policy finds the 0.0.0.0/0 default route in inet.0 and accepts it. The policy also rejects all other routes, thereby not sending all BGP routes on the ISP router. The customer-2-peer policy is for Customer 2 and contains the same policy terms, which also send the default route and no other transit BGP routes. The additional terms in the customer-2-peer policy send the ISP customer routes to Customer 2. Because there are local static routes on ISP Router 3 that represent local customers, these routes are sent as well as all other internal (192.168.0/17) routes announced to the local router by the other ISP routers.

```
[edit]
routing-options {
    static {                                    # simulate local customer routes
        route 192.168.72.0/22 reject;
        route 192.168.76.0/22 reject;
        route 192.168.80.0/22 reject;
        route 192.168.84.0/22 reject;
        route 192.168.88.0/22 reject;
        route 192.168.92.0/22 reject;
        route 192.168.72.0/21 reject;
        route 192.168.80.0/21 reject;
        route 192.168.88.0/21 reject;
    }
    generate {                                  # install a default route if certain routes
                                                from the Exchange Peers are advertised using BGP
        route 0.0.0.0/0 policy if-upstream-routes-exist;
    }
    autonomous-system 1000;
}
```

```
protocols {
    bgp {
        group Internal-Peers {
            type internal;
            local-address 192.168.0.3;
            export internal-peers;
            neighbor 192.168.0.1;
            neighbor 192.168.0.2;
        }
        group Customer-2 {
            type external;
            export customer-2-peer;
            peer-as 2468;
            neighbor 10.222.61.2;
        }
        group Customer-1 {
            type external;
            export customer-1-peer;
            peer-as 1234;
            neighbor 10.222.70.1;
        }
    }
    isis {
        level 1 disable;
        interface so-0/0/0.0;
        interface ge-0/1/0.0;
        interface lo0.0;
    }
}
policy-options {
    policy-statement internal-peers {          # advertise local customer routes to all peers
        term statics {
            from protocol static;
            then accept;
        }
        term next-hop-self {                       # set the BGP next hop to Self for EBGP
                                                routes advertised to IBGP peers
            then {
                next-hop self;
            }
        }
    }
    policy-statement if-upstream-routes-exist {
        term only-certain-contributing-routes {
            from {                                 # allow either the 10.100/17 or the 20.100.0.0 route
                                                   to activate the generated route
                route-filter 10.100.0.0/17 exact;
                route-filter 20.100.0.0/17 exact;
            }
            then accept;
        }                                             # do not allow any other route to activate
                                                   the generated route in the routing table
        term reject-all-other-routes {
            then reject;
        }
    }
```

```
    policy-statement customer-2-peer {          # advertise local customer routes to all peers
        term statics {
            from protocol static;
            then accept;
        }
        term isp-and-customer-routes {          # advertise internal AS 1000 customer routes to the
                                                  customer
            from {
                protocol bgp;
                route-filter 192.168.0.0/17 orlonger;
            }
            then accept;
        }                                           # advertise just the default route to AS 2468
        term default-route {
            from {
                route-filter 0.0.0.0/0 exact;
            }
            then accept;
        }
        term reject-all-other-routes {              # do not advertise any other routes,
                                                    including BGP transit routes
            then reject;
        }
    }
    policy-statement customer-1-peer {
        term default-route {                        # advertise just the default route to AS 1234
            from {
                route-filter 0.0.0.0/0 exact;
            }
            then accept;
        }
        term reject-all-other-routes {              # do not advertise any other routes,
                                                    including BGP transit routes
            then reject;
        }
    }
}
```

Configure Private and Exchange Peers on ISP Routers 1 and 2

ISP Router 1 and ISP Router 2 each have two policies configured: the private-peers policy and the exchange-peers policy. Because of their similar configurations, this example describes the configuration for only ISP Router 2.

On ISP Router 2, the private-peers policy sends the ISP customer routes to the Private Peer 2 router. The policy accepts all local static routes (local ISP Router 2 customers) and all BGP routes in the 192.168.0/17 range (advertised by other ISP routers). These two terms represent the ISP customer routes. The final term rejects all other routes, which includes the entire Internet routing table sent by the exchange peers. These routes do not need to be sent to Private Peer 2 for two reasons:

- The peer already maintains a connection to Exchange Peer 2 in our example, so the routes are redundant.
- The Private Peer wants customer routes only. The private-peers policy accomplishes this goal. The exchange-peers policy sends routes to the Exchange Peer 2 router.

In the example, only two routes need to be sent to Exchange Peer 2:

- The aggregate route that represents the AS 1000 routing space of 192.168.0/17. This route is configured as an aggregate route locally and is advertised by the exchange-peers policy.
- The address space assigned to Customer 2, 192.168.64/22. This smaller aggregate route needs to be sent to Exchange Peer 2 because the customer is also attached to the AS 8000 peer (Private Peer 2).

Sending these two routes to Exchange Peer 2 allows other networks in the Internet to reach the customer through either the ISP or the Private Peer. If just the Private Peer were to advertise the /22 network while the ISP maintained only its /17 aggregate, then all traffic destined for the customer would transit AS 8000 only. Because the customer also wants routes from the ISP, the 192.168.64/22 route is announced by ISP Router 2. Like the larger aggregate route, the 192.168.64/22 route is configured locally and is advertised by the exchange-peers policy. The final term in that policy rejects all routes, including the specific customer networks of the ISP, the customer routes from Private Peer 1, the customer routes from Private Peer 2, and the Internet routing table from Exchange Peer 1. In essence, this final term prevents the ISP from performing transit services for the Internet at large.

```
[edit]
routing-options {
    static {
        route 192.168.32.0/22 reject;
        route 192.168.36.0/22 reject;
        route 192.168.40.0/22 reject;
        route 192.168.44.0/22 reject;
        route 192.168.48.0/22 reject;
        route 192.168.52.0/22 reject;
        route 192.168.32.0/21 reject;
        route 192.168.40.0/21 reject;
        route 192.168.48.0/21 reject;
    }
    aggregate {
        route 192.168.0.0/17;
        route 192.168.64.0/22;
    }
    autonomous-system 1000;
}
protocols {
    bgp {
        group Internal-Peers {
            type internal;
            local-address 192.168.0.2;
            export internal-peers;
            neighbor 192.168.0.1;
            neighbor 192.168.0.3;
        }
        group AS8000-Peers {
            type external;
            export private-peers;
            peer-as 8000;
            neighbor 10.222.45.2;
        }
```

```
        group AS22222-Peers {
            type external;
            export exchange-peers;
            peer-as 22222;
            neighbor 10.222.46.1;
        }
    }
    isis {
        level 1 disable;
        interface so-0/0/0.0;
        interface ge-0/2/0.0;
        interface lo0.0;
    }
}
policy-options {
    policy-statement internal-peers {
        term statics {
            from protocol static;
            then accept;
        }
        term next-hop-self {
            then {
                next-hop self;
            }
        }
    }
    policy-statement private-peers {
        term statics {
            from protocol static;
            then accept;
        }
        term isp-and-customer-routes {
            from {
                protocol bgp;
                route-filter 192.168.0.0/17 orlonger;
            }
            then accept;
        }
        term reject-all {
            then reject;
        }
    }
    policy-statement exchange-peers {
        term AS1000-Aggregate {
            from {
                protocol aggregate;
                route-filter 192.168.0.0/17 exact;
            }
            then accept;
        }

        term Customer-2-Aggregate {
            from {
                protocol aggregate;
                route-filter 192.168.64.0/22 exact;
            }
            then accept;
        }
        term reject-all-other-routes {
            then reject;
        }
    }
}
```

Configure Locally Defined Static Routes on the Exchange Peer 2 Router

The Exchange Peer 2 router exchanges all routes with all BGP peers. The outbound-routes policy for Exchange Peer 2 advertises locally defined static routes using BGP.

```
[edit]
protocols {
    bgp {
        group Peers {
            type external;
            export outbound-routes;
            neighbor 10.222.4.1 {
                peer-as 11111;
            }
            neighbor 10.222.44.2 {
                peer-as 8000;
            }
            neighbor 10.222.46.2 {
                peer-as 1000;
            }
        }
    }
}
policy-options {
    policy-statement outbound-routes {        # advertise the simulated Internet routes to all BGP
                                               peers
        term statics {
            from protocol static;
            then accept;
        }
    }
}
```

Configure Outbound and Generated Routes on the Private Peer 2 Router

The Private Peer 2 router performs two main functions:

- Advertise routes local to AS 8000 to both the Exchange Peers and the ISP routers. The outbound-routes policy advertises the local static routes (that is, customers) on the router, and also advertises all routes learned by BGP that originated in either AS 8000 or AS 2468. These routes include other AS 8000 customer routes in addition to the AS 2468 customer. The AS routes are identified by an AS path regular expression match criteria in the policy.

- Advertise the 0.0.0.0/0 default route to the AS 2468 customer router. To accomplish this, the Private Peer creates a generated route for 0.0.0.0/0 locally on the router. This generated route is further assigned a policy called if-upstream-routes-exist, which allows only certain routes to contribute to the generated route, making it an active route in the routing table. Once the route is active, it can be sent to the AS 2468 router using BGP and the configured policies. The if-upstream-routes-exist policy accepts only the 20.100.0.0/17 route from Exchange Peer 2, and rejects all other routes. If the 20.100.0.0/17 route is withdrawn by the Exchange Peer, the Private Peer loses the 0.0.0.0/0 default route and withdraws the default route from the AS 2468 customer router.

```
[edit]
routing-options {                       # simulate local customer routes
   static {
      route 172.16.64.0/20 reject;
      route 172.16.80.0/20 reject;
      route 172.16.96.0/20 reject;
      route 172.16.112.0/20 reject;
      route 172.16.72.0/21 reject;
      route 172.16.88.0/21 reject;
      route 172.16.104.0/21 reject;
      route 172.16.120.0/21 reject;
   }
generate {                              # install a default route if certain routes from
                                        the Exchange Peer are advertised using BGP
      route 0.0.0.0/0 policy if-upstream-routes-exist;
   }
   autonomous-system 8000;
}
protocols {
   bgp {
      group External-Peers {
         type external;
         export outbound-routes;
         neighbor 10.222.44.1 {
            peer-as 22222;
         }
         neighbor 10.222.45.1 {
            peer-as 1000;
         }
      }
      group Customers {
         type external;
         export internal-routes;
         neighbor 10.222.6.1 {
            peer-as 2468;
         }
      }
   }
}
policy-options {
   policy-statement outbound-routes {      # advertise local customer routes to
                                           AS 1000 and the Exchange Peers
      term statics {
         from protocol static;
         then accept;
      }
      term allowed-bgp-routes {
         from {                           # advertise routes originated in AS 2468 and AS 8000
                                           to AS 1000 and the Exchange Peers
            protocol bgp;
            as-path [ my-own-routes AS2468-routes ];
         }
         then accept;
      }
      term no-transit {
         then reject;                             # do not advertise any other routes,
                                                  including BGP transit routes
      }
   }
```

```
    policy-statement internal-routes {          # advertise local customer routes to AS 2468
        term statics {
            from protocol static;
            then accept;
        }
        term default-route {                    # advertise just the default route to AS 2468
            from {
                route-filter 0.0.0.0/0 exact;
            }
            then accept;
        }
        term reject-all-other-routes {          # do not advertise any other routes,
                                                including BGP transit routes
            then reject;
        }
    }
    policy-statement if-upstream-routes-exist {
        term as-22222-routes {
            from {                              # allow the 20.100.0.0 route to activate
                                                   the generated route in the routing table
                route-filter 20.100.0.0/17 exact;
            }
            then accept;
        }
        term reject-all-other-routes {
            then reject;                        # do not allow any other route to activate
                                                   the generated route in the routing table
        }
    }
    as-path my-own-routes "()";
    as-path AS2468-routes “2468”;
}
```

Configure the Discard Interface

The discard interface allows you to protect a network from denial-of-service (DoS) attacks by identifying the target IP address that is being attacked and configuring a policy to forward all packets to a discard interface. All packets forwarded to the discard interface are dropped.

To configure the discard interface, include the dsc statement at the [edit interface] hierarchy level:

```
[edit]
interface {
    dsc {
        unit 0 {
            family inet {
                filter {
                    input filter-name;
                    output filter-name;
                }
            }
        }
    }
}
```

The dsc interface name denotes the discard interface. The discard interface supports only unit 0. For more information about configuring interfaces, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

The following two configurations are required to configure a policy to forward all packets to the discard interface.

Configure an input policy to associate a community with the discard interface:

```
[edit]
policy-options {
    community community-name members [ community-id ];
    policy-statement statement-name {
        term term-name {
            from community community-name;
            then next-hop dsc;
        }
    }
}
```

Configure an output policy to set up the community on the routes injected into the network:

```
[edit]
policy-options {
    policy-statement statement-name {
        term term-name {
            from prefix-list name;
            then community (set | add | delete) community-name;
        }
    }
}
```

Test Routing Policies

Before applying a routing policy in a live environment, you can use the test policy command to ensure that the policy produces the results that you expect. The test policy command uses the following syntax:

```
test policy policy-name prefix
```

Example: Test a Policy

Test the following policy, which looks for unwanted routes and rejects them:

```
[edit policy-options]
policy-statement reject-unwanted-routes {
    term drop-these-routes {
        from {
            route-filter 0/0 exact;
            route-filter 127/8 orlonger;
            route-filter 10/8 orlonger;
            route-filter 172.16/12 orlonger;
            route-filter 192.168/16 orlonger;
            route-filter 224/3 orlonger;
         }
        then reject;
    }
}
```

Test this policy against all routes in the routing table:

```
user@host> test policy reject-unwanted-routes 0/0
```

Test this policy against a specific set of routes:

```
user@host> test policy reject-unwanted-routes 10.49.0.0/16
```

Chapter 5

Configure Extended Match Conditions

This chapter describes how to configure extended match conditions for a routing policy. In general, the extended match conditions include criteria that are defined separately from the routing policy (autonomous system [AS] path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. You should have an in-depth understanding of extended match conditions before configuring them. For information about standard match conditions, see Table 9 on page 41.

This chapter describes how to configure the following extended match conditions:

- Configure AS Path Regular Expressions on page 83
- Configure Communities on page 90
- Configure Prefix Lists on page 102
- Configure Route Lists on page 106
- Configure Subroutines on page 114

Configure AS Path Regular Expressions

A Border Gateway Protocol (BGP) *AS path* is a path to a destination. An AS path consists of AS numbers of networks that a packet traverses if it takes the associated route to a destination. The AS numbers are assembled in a sequence from left to right, for example, AS5, AS4, AS3, AS2, AS1. For a packet to reach a destination using this route, it first traverses AS5 and so on until it reaches AS1, which is the last AS before its destination.

You can define a match condition based on all or portions of the AS path. To do this, you create a named AS path regular expression and then include it in a routing policy.

This section describes the following tasks for configuring AS path regular expressions and provides the following examples:

- Define AS Path Regular Expressions on page 84
- How AS Path Regular Expressions Are Evaluated on page 88
- Examples: Configure AS Path Regular Expressions on page 89

Define AS Path Regular Expressions

You can create a named AS path regular expression and then include it in a routing policy with the as-path match condition (described in Table 9 on page 41). To create a named AS path regular expression, include the as-path statement at the [edit policy-options] hierarchy level:

```
[edit policy-options]
as-path name regular-expression;
```

To include the AS path regular expression in a routing policy, include the as-path match condition in the from statement:

```
[edit policy-options]
as-path name regular-expression;
policy-statement policy-name {
    term term-name {
        from {
            as-path names;
        }
    }
}
```

Additionally, you can create a named AS path group made up of AS path regular expressions and then include it in a routing policy with the as-path-group match condition. To create a named AS path group, include the as-path-group statement at the [edit policy-options] hierarchy level:

```
[edit policy-options]
as-path-group group-name {
    [ as-path name regular-expression ];
}
```

To include the AS path regular expressions within the AS path group in a routing policy, include the as-path-group match condition in the from statement:

```
[edit policy-options]
as-path-group group-name {
    [ as-path name regular-expression ];
}
policy-statement policy-name {
    term term-name {
        from {
            as-path-group group-name;
        }
    }
}
```

You cannot have both as-path and as-path-group in the same policy term.

You can include the names of multiple AS path regular expressions in the as-path match condition in the from statement. If you do this, only one AS path regular expression needs to match for a match to occur. The AS path regular expression matching is effectively a logical OR operation.

The AS path name identifies the regular expression. It can contain letters, numbers, and hyphens (–), and can be up to 255 characters. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

The regular expression is used to match all or portions of the AS path. It consists of two components, which you specify in the following format:

term <operator>

- *term*—Identifies an AS. You can specify it in one of the following ways:
  - AS number—The entire AS number composes one *term*. You cannot reference individual characters within an AS number, which differs from regular expressions as defined in POSIX 1003.2.
  - Wildcard character—Matches any single AS number. The wildcard character is a period (.). You can specify multiple wildcard characters.
  - AS path—A single AS number or a group of AS numbers enclosed in parentheses. Grouping the regular expression in this way allows you to perform a common operation on the group as a whole and to give the group precedence. The grouped path can itself include operators.
- *operator*—(Optional) An operator specifying how the term must match. Most operators describe how many times the term must be found to be considered a match (for example, any number of occurrences, or zero or one occurrence). Table 14 lists the regular expression operators supported for AS paths. You place operators immediately after *term* with no intervening space, except for the pipe (|) and dash (–) operators, which you place between two terms, and parentheses, with which you enclose terms.

You can specify one or more term–operator pairs in a single regular expression.

Table 15 shows examples of how to define regular expressions to match AS paths.

Table 14: AS Path Regular Expression Operators

Operator	Match...
{*m*,*n*}	At least *m* and at most *n* repetitions of *term*. Both *m* and *n* must be positive integers, and *m* must be smaller than *n*.
{*m*}	Exactly *m* repetitions of *term*. *m* must be a positive integer.
{*m*,}	*m* or more repetitions of *term*. *m* must be a positive integer.
*	Zero or more repetitions of *term*. This is equivalent to {0,}.
+	One or more repetitions of *term*. This is equivalent to {1,}.
?	Zero or one repetition of *term*. This is equivalent to {0,1}.
|	One of the two terms on either side of the pipe.
–	Between a starting and ending range, inclusive.
^	Character at the beginning of an AS path regular expression. This character is added implicitly; therefore, the use of it is optional.
$	Character at the end of an AS path regular expression. This character is added implicitly; therefore, the use of it is optional.
( )	A group of terms that are enclosed in the parentheses. If enclosed in quotation marks with no intervening space (“()”), indicates a null. Intervening space between the parentheses and the terms is ignored.
[ ]	Set of characters. One character from the set can match. To specify the start and end of a range, use a hyphen (–).

Table 15: Examples of Defining AS Path Regular Expressions

AS Path to Match	Regular Expression	Example Matches
AS path is 1234	1234	1234
Zero or more occurrences of AS number 1234	1234*	1234 1234 1234 1234 1234 1234 Null AS path
Zero or one occurrence of AS number 1234	1234? or 1234{0,1}	1234 Null AS path
One through four occurrences of AS number 1234	1234{1,4}	1234 1234 1234 1234 1234 1234 1234 1234 1234 1234
One through four occurrences of AS number 12 followed by one occurrence of AS number 34	12{1,4} 34	12 34 12 12 34 12 12 12 34 12 12 12 12 34
Range of AS numbers to match a single AS number	123–125 [123–125]*	123 or 124 or 125 Null AS path 123 124 124 125 125 125
Path whose second AS number must be 56 or 78	(. 56) | (. 78) or . (56|78)	1234 56 34 78
Path whose second AS number might be 56 or 78	. (56|78)?	1234 56 34
Path whose first AS number is 123 and second AS number is either 56 or 78	123 (56|78)	123 56 123 78
Path of any length, except nonexistent, whose second AS number can be anything, including nonexistent	..* or ..{0,}	1234 1234 5678 1234 5 6 7 8
AS path is 1 2 3	1 2 3	1 2 3
One occurrence of the AS numbers 1 and 2, followed by one or more occurrences of the number 3	1 2 3 + ;	1 2 3 1 2 3 3 1 2 3 3 3
One or more occurrences of AS number 1, followed by one or more occurrences of AS number 2, followed by one or more occurrences of AS number 3	1 + 2 + 3 +	1 2 3 1 1 2 3 1 1 2 2 3 1 1 2 2 3 3
Path of any length that begins with AS numbers 4, 5, 6	4 5 6 .*	4 5 6 4 5 6 7 8 9
Path of any length that ends with AS numbers 4, 5, 6	.* 4 5 6	4 5 6 1 2 3 4 5 6
AS path 5, 12, or 18	[5 12 18]	5 12 18

Null AS Path

You can use AS path regular expressions to create a null AS path that matches routes (prefixes) that have originated in your AS. These routes have not been advertised to your AS by any external peers. To create a null AS path, use the parentheses operator enclosed in quotation marks with no intervening spaces:

```
"()"
```

Example: Null AS Path

AS 1 and AS 3 are connected to AS 2, which you administrate. AS 3 advertises its routes to your AS, but you do not want to advertise AS 3 routes to AS 1 and thereby begin routing traffic from AS 1 to AS 3 through your AS. To prevent this situation from occurring, you can configure an export policy for AS 1 (1.2.2.6) that allows routes for your AS to be advertised to AS 1 but does not allow routes for AS 3 or routes for any other connected AS to be advertised to AS 1:

```
[edit policy-options]
as-path null-as "()";
policy-statement only-my-routes {
    term just-my-as {
        from {
            protocol bgp;
            as-path null-as;
        }
        then accept;
    }
    term nothing-else {
        then reject;
    }
}
protocol {
    bgp {
        neighbor 1.2.2.6 {
            export only-my-routes;
        }
    }
}
```

How AS Path Regular Expressions Are Evaluated

AS path regular expressions implement the extended (modern) regular expressions as defined in POSIX 1003.2 They are identical to the UNIX regular expressions with the following exceptions:

- The basic unit of matching in an AS path regular expression is the AS number and not an individual character.
- A regular expression matches a route only if the AS path in the route exactly matches *regular-expression*. The equivalent UNIX regular expression is ^*regular-expression*$. For example, the AS path regular expression 1234 is equivalent to the UNIX regular expression ^1234$.
- You can specify a regular expression using wildcard operators.

Examples: Configure AS Path Regular Expressions

Exactly match routes with the AS path 1234 56 78 9 and accept them:

```
[edit]
policy-options {
    as-path wellington “1234 56 78 9”;
    policy-statement from-wellington {
        term term1 {
            from as-path wellington;
            }
            then {
                preference 200;
                accept;
            }
        term term2 {
            then reject;
        }
    }
}
```

Match alternate paths to an AS and accept them after modifying the preference:

```
[edit]
policy-options {
    as-path wellington-alternate “1234{1,6} (56|47)? (78|101|112)* 9+”;
    policy-statement from-wellington {
        from as-path wellington-alternate;
        }
        then {
            preference 200;
            accept;
        }
    }
}
```

Match routes with an AS path of 123, 124, or 125 and accept them after modifying the preference:

```
[edit]
policy-options {
    as-path addison “123–125”;
    policy-statement from-addison {
        from as-path addison;
        }
        then {
            preference 200;
            accept;
        }
    }
}
```

Configure Communities

A *BGP community* is a group of destinations that share a common property. Community information is included as a path attribute in BGP update messages. This information identifies community members and allows you to perform actions on a group without having to elaborate upon each member. You can define match conditions based on a BGP community, which this section describes. For information about actions that can be performed on the community, see Table 11 on page 45.

You can assign community tags to non-BGP routes through configuration (for static, aggregate, or generated routes) or an import routing policy. These tags can then be matched when BGP exports the routes.

This section includes the following information:

Define Communities

You can create a named community and include it in a routing policy with the community match condition (described in Table 9 on page 41).

You can configure community and extended communities attributes to be included in BGP update messages. The community attribute is only four octets. The BGP extended communities attribute provides a larger range (eight octets) for grouping or categorizing communities. You can use community and extended communities attributes to trigger routing decisions, such as acceptance, rejection, preference, or redistribution.

The *community-ids* format varies according to the type of attribute that you use. The BGP community attribute format is *as-number:community-value*. The BGP extended communities attribute format is *type:administrator:assigned-number*.

When specifying *community-ids* for the community attribute, you can use UNIX-style regular expressions. Regular expressions are not supported for the extended communities attributes.

To define communities, you can do the following:

Configure the Community Attribute

To create a named community and define the community members, include the community statement at the [edit policy-options] hierarchy level:

```
[edit policy-options]
community name {
    invert-match;
    members [ community-ids ];
}
```

To include the community in a routing policy, include the community condition in the from statement:

```
[edit policy-options]
community name members [ community-ids ];
policy-statement policy-name {
    term term-name {
        from {
            community names;
        }
    }
}
```

You can include the names of multiple communities in the community match condition in the from statement. If you do this, only one community needs to match for a match to occur. The community matching is effectively a logical OR operation.

name identifies the community or communities. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

community-ids defines one or more members of the community. It consists of two components, which you specify in the following format:

as-number:community-value

- *as-number*—AS number of the community member. It can be a value from 1 through 65,534. You can specify the AS number in one of the following ways:
  - AS number.
  - Asterisk (*)—A wildcard character that matches all AS numbers. (In the definition of the community attribute, the asterisk also functions as described in Table 16 on page 94.)
  - Period (.)—A wildcard character that matches any single digit in an AS number.
  - Group of AS numbers—A single AS number or a group of AS numbers enclosed in parentheses. Grouping the numbers in this way allows you to perform a common operation on the group as a whole and to give the group precedence. The grouped numbers can themselves include regular expression operators. For more information about the community regular expressions, see “Configure the Community Attribute Using UNIX Regular Expressions” on page 93.

- *community-value*—Identifier of the community member. It can be a number from 0 through 65,535. You can specify the community value in one of the following ways:
  - Community value number.
  - Asterisk (*)—A wildcard character that matches all community values. (In the definition of the community attribute, the asterisk also functions as described in Table 16 on page 94.)
  - Period (.)—A wildcard character that matches any single digit in a community value number.
  - Group of community value numbers—A single community value number or a group of community value numbers enclosed in parentheses. Grouping the regular expression in this way allows you to perform a common operation on the group as a whole and to give the group precedence. The grouped path can itself include regular expression operators.

You can use the community delete action with any defined community member, including those defined using the wildcard character (*). You can use the community add or community set actions with any defined community member except those defined using the wildcard character.

You also can specify *community-id* as one of the following well-known community names, which are defined in Request for Comments (RFC) 1997:

- no-advertise—Routes in this community name must not be advertised to other BGP peers.
- no-export—Routes in this community must not be advertised outside a BGP confederation boundary.
- no-export-subconfed—Routes in this community must not be advertised to external BGP peers, including peers in other members' ASs inside a BGP confederation.

Additionally, you can explicitly exclude BGP community information with a static route by using the none option. Include this option when configuring an individual route in the route portion to override a community option specified in the defaults portion.

See also "Configure the Extended Communities Attribute" on page 99.

Configure the Community Attribute Using UNIX Regular Expressions

When specifying *community-ids*, you can use UNIX-style regular expressions to specify the AS number and the member identifier. A regular expression consists of two components, which you specify in the following format:

term<operator>

term identifies the string to match.

operator specifies how the term must match. Table 16 lists the regular expression operators supported for the community attribute. You place an operator immediately after *term* with no intervening space, except for the pipe (|) and dash (–) operators, which you place between two terms, and parentheses, with which you enclose terms. Table 17 shows examples of how to define *community-ids* using community regular expressions. The operator is optional.

Community regular expressions are identical to the UNIX regular expressions. Both implement the extended (or modern) regular expressions as defined in POSIX 1003.2.

Community regular expressions evaluate the string specified in *term* on a character-by-character basis. For example, if you specify 1234:5678 as *term*, the regular expressions see nine discrete characters, including the colon (:), instead of two sets of numbers (1234 and 5678) separated by a colon.

Table 16: Community Attribute Regular Expression Operators

Operator	Match...
{*m*,*n*}	At least *m* and at most *n* repetitions of *term*. Both *m* and *n* must be positive integers, and *m* must be smaller than *n*.
{*m*}	Exactly *m* repetitions of *term*. *m* must be a positive integer.
{*m*,}	*m* or more repetitions of *term*. *m* must be a positive integer.
*	Zero or more repetitions of *term*. This is equivalent to {0,}.
+	One or more repetitions of *term*. This is equivalent to {1,}.
?	Zero or one repetition of *term*. This is equivalent to {0,1}.
|	One of the two terms on either side of the pipe.
–	Between a starting and ending range, inclusive.
^	Character at the beginning of a community attribute regular expression. We recommend the use of this operator for the clearest interpretation of your community attribute regular expression. If you do not use this operator, the regular expression 123:456 could also match a route tagged with 5123:456.
$	Character at the end of a community attribute regular expression. We recommend the use of this operator for the clearest interpretation of your community attribute regular expression. If you do not use this operator, the regular expression 123:456 could also match a route tagged with 123:4563.
[ ]	Set of characters. One character from the set can match. To specify the start and end of a range, use a hyphen (–). To specify a set of characters that do not match, use the caret (^) as the first character after the opening square bracket ([).
( )	A group of terms that are enclosed in the parentheses. If enclosed in quotation marks with no intervening space (“()”), indicates a null. Intervening space between the parentheses and the terms is ignored.

Table 17: Examples of Defining Community Attribute Regular Expressions

Community Attribute to Match	Regular Expression	Example Matches
AS number is 56 or 78. Community value is any number.	^((56) | (78)):(.*)$	56:1000 78:65000
AS number is 56. Community value is any number that starts with 2.	^56:(2.*)$	56:2 56:222 56:234
AS number is any number. Community value is any number that ends with 5, 7, or 9.	^(.*):(.*[579])$	1234:5 78:2357 34:65009
AS number is 56 or 78. Community value is any number that starts with 2 and ends with 2 through 8.	^((56) | (78)):(2.*[2–8])$	56:22 56:21197 78:2678

Do Not Advertise Communities to Neighbors

By default, communities are sent to BGP peers. To suppress the advertisement of communities to a neighbor, remove all communities. When the result of an export policy is an empty set of communities, the community attribute is not sent. To remove all communities, first define a wildcard set of communities (here, the community is named wild):

```
[edit policy-options]
community wild members "*:*";
```

Then, in the routing policy statement, specify the community delete action:

```
[edit policy-options]
policy-statement policy-name {
    term term-name {
        then community delete wild;
    }
}
```

To suppress a particular community from any AS, define the community as community wild members "*:*community-value*".

Examples: Configure the Community Attribute

Create a community named dunedin and apply it in a routing policy statement:

```
[edit]
policy-options {
    community dunedin members [56:2379 23:46944];
    policy-statement from-dunedin {
        from community dunedin;
        then {
            metric 2;
            preference 100;
            next policy;
        }
    }
}
```

The above example modifies the metric and preference for routes that contain members of community dunedin only.

Delete a particular community from a route, leaving remaining communities untouched:

```
[edit]
policy-options {
    community dunedin members 701:555;
    policy-statement delete-dunedin {
        then {
            community delete dunedin;
        }
    }
}
```

Remove any community from a route with the AS number of 65534 or 65535:

```
[edit]
policy-options {
    community my-as1-transit members [65535:10 65535:11];
    community my-as2-transit members [65534:10 65534:11];
    community my-wild members [65534:* 65535:*];
    policy-statement delete-communities {
        from {
            community [my-as1-transit my-as2-transit];
        }
        then {
            community delete my-wild;
        }
    }
}
```

Match the set of community members 5000, 5010, 5020, 5030, and so on up to 5090:

```
[edit]
policy-options {
    community customers members "^1111:50.0$"
    policy-statement advertise-customers {
        from community customers;
        then accept;
    }
}
```

Reject routes that are longer than /19 in Class A space, /16 in Class B space, and /24 in Class C space:

```
[edit policy-options]
community auckland-accept members 555:1;
policy-statement drop-specific-routes {
    from {
        route-filter 0.0.0.0/1 upto /19 {
            community add auckland-accept
            next policy;
        }
        route-filter 128.0.0.0/2 upto /16 {
            community add auckland-accept
            next policy;
        }
        route-filter 192.0.0.0/3 upto /24 {
            community add auckland-accept
            next policy;
        }
    }
    then reject;
}
```

In the above example, for routes that are not rejected, the tag auckland-accept is added.

Create routing policies to handle peer and customer communities. This example does the following:

- Customer routes that match the attributes defined in the lcl20*x*-low communities, for example, lcl201-low, are accepted and their local preference is changed to 80.
- Customer routes that match the attributes defined in the lcl20*x*-high communities, for example, lcl201-high, are accepted and have their local preference changed to 120.

- Internal routes that match the attributes defined in the internal20*x* communities, for example, internal201, are rejected and not advertised to customers.

- Routes received from a peer are assigned a metric of 10 and the community defined in peer201.

- Routes that match the attributes defined in the prepend20*x-x* communities, for example, prepend201-1, prepend201-2, or prepend201-3, are sent to peers and have the AS number 201 prepended the specified number of times.

- Routes that match the attributes defined in the peer20*x*, custpeer20*x*, and internal20*x* communities, for example, peer201, custpeer201, or internal201, respectively, are rejected and not advertised to peers.

```
[edit]
policy-options {
    community internal201 members 201:112;
    community internal202 members 202:112;
    community internal203 members 203:112;
    community internal204 members 204:112;
    community internal205 members 205:112;
    community peer201 members 201:555;
    community peer202 members 202:555;
    community peer203 members 203:555;
    community peer204 members 204:555;
    community peer205 members 205:555;
    community custpeer201 members 201:20;
    community custpeer202 members 202:20;
    community custpeer203 members 203:20;
    community custpeer204 members 204:20;
    community custpeer205 members 205:20;
    community prepend201-1 members 201:1;
    community prepend202-1 members 202:1;
    community prepend203-1 members 203:1;
    community prepend204-1 members 204:1;
    community prepend205-1 members 205:1;
    community prepend201-2 members 201:2;
    community prepend202-2 members 202:2;
    community prepend203-2 members 203:2;
    community prepend204-2 members 204:2;
    community prepend205-2 members 205:2;
    community prepend201-3 members 201:3;
    community prepend202-3 members 202:3;
    community prepend203-3 members 203:3;
    community prepend204-3 members 204:3;
    community prepend205-3 members 205:3;
    community lcl201-low members 201:80;
    community lcl202-low members 202:80;
    community lcl203-low members 203:80;
    community lcl204-low members 204:80;
    community lcl205-low members 205:80;
```

```
community lcl 20x-high members "^20 [ 1-5 ] : 120$";
policy-statement in-customer {
    term term1 {
        from {
            protocol bgp;
            community lcl 20x-high;
        }
        then {
            local-preference 80;
            accept;
        }
    }
    term term2 {
        from {
            protocol bgp;
            community [lcl201-high lcl202-high lcl203-high lcl204-high lcl205-high];
        }
        then local-preference 120;
    }
    then next policy;
}
policy-statement out-customer {
    term term1 {
        from {
            protocol bgp;
            community [internal201 internal202 internal203 internal204 internal205];
        }
        then reject;
    }
    then next policy;
}
policy-statement in-peer {
    from protocol bgp;
    then {
        metric 10;
        community set peer201;
    }
}
policy-statement out-peer {
    term term1{
        from {
            protocol bgp;
            community [prepend201-1 prepend202-1 prepend203-1 prepend204-1
                prepend205-1];
        }
        then as-path-prepend 201;
    }
    term term2 {
        from {
            protocol bgp;
            community [prepend201-2 prepend202-2 prepend203-2 prepend204-2
                prepend205-2];
        }
        then as-path-prepend "201 201";
    }
```

```
        term term3 {
            from {
                protocol bgp;
                    community [prepend201-3 prepend202-3 prepend203-3
                        prepend204-3 prepend205-3];
            }
            then as-path-prepend “201 201 201”;
        }
        term term4 {
            from {
                protocol bgp;
                community [peer201 peer202 peer203 peer204 peer205 custpeer201
                    custpeer202 custpeer203 custpeer204 custpeer205 internal201
                    internal202 internal203 internal204 internal205];
            }
            then reject;
        }
        then next policy;
    }
}
```

Configure the Extended Communities Attribute

To configure extended communities, include the community statement at the [edit policy-options] hierarchy level:

```
[edit policy-options]
community name {
    invert-match;
    members [ community-ids ];
}
```

To include the community in a routing policy, include the community condition in the from statement:

```
[edit policy-options]
community name members [ community-ids ];
policy-statement policy-name {
    term term-name {
        from {
            community name;
        }
    }
}
```

name identifies one or more routers in the BGP extended community.

community-ids identifies the type of extended community in the following format:

```
type:administrator:assigned-number
```

type is the type of extended community and can be either a bandwidth, target, origin, or domain-id community. The bandwidth community sets up the bandwidth extended community. The target community identifies the destination to which the route is going. The origin community identifies where the route originated. The domain-id community identifies the Open Shortest Path First (OSPF) domain from which the route originated.

administrator is the administrator. It is either an AS number or an Internet Protocol Version 4 (IPv4) address prefix, depending on the type of extended community.

assigned-number identifies the local provider.

For more information about BGP extended communities, see *BGP Extended Communities Attribute*, Internet draft draft-ramachandra-bgp-ext-communities-04.txt. For information about the community attribute, see “Configure the Community Attribute” on page 91.


Note

Regular expressions are not supported for extended communities.

Examples: Configure the Extended Communities Attribute

Configure a target community with an administrative field of 10458 and an assigned number of 20:

```
[edit]
policy-options {
    community test-a members [target:10458:20];
}
```

Configure a target community with an administrative field of 1.1.1.1 and an assigned number of 20:

```
[edit]
policy-options {
    community test-a members [target:1.1.1.1:20];
}
```

Configure an origin community with an administrative field of 1.1.1.1 and an assigned number of 20:

```
[edit]
policy-options {
    community test-a members [origin:1.1.1.1:20];
}
```

Invert Community Matches

To invert the results of the community expression matching, include the invert-match statement at the [edit policy-options community *name*] hierarchy level:

```
[edit policy-options community name]
invert-match;
```

Configure Link Bandwidth

To configure the link bandwidth extended communities attribute, include the bandwidth statement at the [edit policy-options community *name* members] hierarchy level:

```
[edit policy-options community name members]
bandwidth:as:num;
```

Specifying link bandwidth allows you to distribute traffic unequally among different BGP paths.

The link bandwidth attribute does not work concurrently with per-packet load-balancing.

How Communities Are Evaluated

The policy framework software evaluates communities as follows:

- Each route is evaluated against each named community in a routing policy from statement. If a route matches one of the named communities in the from statement, the evaluation of the current term continues. If a route does not match, the evaluation of the current term ends.
- The route is evaluated against each member of a named community. The evaluation of all members must be successful for the named community evaluation to be successful.
- Each member in a named community is either a literal community value or a regular expression. Each member is evaluated against each community associated with the route. (Communities are an unordered property of a route. For example, 1:2 3:4 is the same as 3:4 1:2.) Only one community from the route is required to match for the member evaluation to be successful.
- Community regular expressions are evaluated on a character-by-character basis. For example, if a route contains community 1234:5678, the regular expressions see nine discrete characters, including the colon (:), instead of two sets of numbers (1234 and 5678) separated by a colon. For example:

```
[edit]
policy-options {
    policy-statement one {
        from {
            community [comm-one comm-two];
        }
    }
    community members comm-one [ 1:2 "^4:(5|6)$" ];
    community members comm-two [ 7:8 9:10 ];
}
```

- To match routing policy one, the route must match either comm-one or comm-two.
- To match comm-one, the route must have a community that matches 1:2 and a community that matches 4:5 or 4:6.
- To match comm-two, the route must have a community that matches 7:8 and a community that matches 9:10.

Configure Prefix Lists

A *prefix list* is a named list of IP addresses. You can specify an exact match with incoming routes and apply a common action to all matching prefixes in the list.

Because the configuration of prefix lists includes setting up prefixes and prefix lengths, we strongly recommend that you have a thorough understanding of IP addressing, including supernetting, before proceeding with the configuration.

This section includes the following information:

Prefix List and Route List Differences

A prefix list functions similarly to a route list that contains multiple instances of the exact match type only. The similarities between these two extended match conditions which are summarized in Table 18.

Table 18: Prefix List and Route List Differences

Feature	Prefix List	Route Lists
Match types	Does not support match types. The specified prefixes must be matched exactly.	Supports several match types. For more information, see Table 19 on page 107.
Action	Can specify action in a then statement only. These actions are applied to all prefixes that match the term.	Can specify action that is applied to a particular prefix in a route-filter match condition in a from statement, or to all prefixes in the list using a then statement.

Define Prefix Lists

You can create a named prefix list and include it in a routing policy with the prefix-list match condition (described in Table 9 on page 41).

To define a prefix list, include the prefix-list statement at the [edit policy-options] hierarchy level:

```
[edit policy-options]
prefix-list name {
    apply-path path;
    ip-addresses;
}
```

You can use the apply-path statement to include all prefixes pointed to by a defined path, or you can specify one or more addresses, or both.

To include a prefix list in a routing policy, specify the prefix-list match condition in the from statement:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                prefix-list name;
            }
            then actions;
        }
    }
}
```

name identifies the prefix list. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

ip-addresses are the Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) prefixes specified as *prefix*/*prefix-length*. If you omit *prefix-length* for an IPv4 prefix, the default is /32. If you omit *prefix-length* for an IPv6 prefix, the default is /128. Prefixes specified in a from statement must be either all IPv4 addresses or all IPv6 addresses.

You cannot apply actions to individual prefixes in the list.

You can specify the same prefix list in the from statement of multiple routing policies or firewall filters. For information about firewall filters, see “Firewall Filters” on page 137.

Use the apply-path statement to configure a prefix list comprising of all IP prefixes pointed to by a defined path. This eliminates most of the effort required to maintain a group prefix list.

The path consists of elements separated by spaces. Each element matches a configuration keyword or an identifier, and you can use wildcards to match more than one identifier. Wildcards must be enclosed in angle brackets, for example, <*>.

When you use apply-path to define a prefix list, the prefix list can be used in a firewall filter only. It cannot be used in a policy statement.

See “Example: Configure a Prefix List” on page 105, and see “Configure Firewall Filters” for using a prefix list in a firewall filter.

How a Prefix List Is Evaluated

During prefix list evaluation, the policy framework software performs a *longest-match lookup*, which means that the software searches for the prefix in the list with the longest length. The order in which you specify the prefixes, from top to bottom, does not matter. The software then compares a route's source address to the longest prefix.

If a match occurs, the evaluation of the current term continues. If a match does not occur, the evaluation of the current term ends.

If you specify multiple prefixes in the prefix list, only one prefix must match for a match to occur. The prefix list matching is effectively a logical OR operation.

Example: Configure a Prefix List

The following example accepts and rejects traffic from sites specified using prefix lists:

```
[edit]
policy-options {
    policy-statement prefix-list-policy {
        term ok-sites {
            from {
                prefix-list known-ok-sites
            }
            then accept;
        }
        term reject-bcasts {
            from {
                prefix-list known-dir-bcast-sites
            }
            then reject;
        }
    }
}

[edit]
policy-options {
    prefix-list known-ok-sites {
        172.8.0.3;
        10.10.0.0/16;
        192.168.12.0/24;
    }

[edit]
policy-options {
    prefix-list known-dir-bcast-sites {
        10.3.4.6;
        10.2.0.0/16;
        192.168.1.0/24;
    }
}
```

Configure Route Lists

A *route list* is a collection of destination prefixes. When specifying a prefix, you can specify an exact match with a particular route or a less precise match. You can configure either a common action that applies to the entire list or an action associated with each prefix.

Because the configuration of route lists includes setting up prefixes and prefix lengths, we strongly recommend that you have a thorough understanding of IP addressing, including supernetting, before proceeding with the configuration.

It is also important to understand how a route list is evaluated, particularly if the route list includes multiple route-filter options in a from statement. We strongly recommend that you read "How a Route List Is Evaluated" on page 108 before proceeding with the configuration. Not fully understanding the evaluation process could result in faulty configuration and unexpected results.

This section discusses the following topics:

Define Route Lists

To specify a route list, include one or more route-filter or source-address-filter options in the from statement of the policy-statement:

```
[edit]
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
            }
            then actions;
        }
    }
}
```

The route-filter option is typically used to match prefixes of any type except for multicast source addresses.

The source-address-filter option is typically used to match multicast source addresses in multiprotocol BGP (MBGP) and Multicast Source Discovery Protocol (MSDP) environments.

destination-prefix is the IPv4 or IPv6 prefix specified as *prefix*/*prefix-length*. If you omit *prefix-length* for an IPv4 prefix, the default is /32. If you omit *prefix-length* for an IPv6 prefix, the default is /128. Prefixes specified in a from statement must be either all IPv4 addresses or all IPv6 addresses.

match-type is the type of match to apply to the destination prefix. It can be one of the match types listed in Table 19. For examples of the match types and the results when presented with various routes, see Table 20.

actions is the action to take if the destination prefix matches. It can be one or more of the actions listed in Table 10 on page 44 and Table 11 on page 45.

In route lists, you can specify actions in two ways:

- In the route-filter or source-address-filter option—These actions are taken immediately after a match occurs, and the then statement is not evaluated.
- In the then statement—These actions are taken after a match occurs and if an action is not specified in the route-filter or source-address-filter option.

The upto and prefix-length-range match types are similar in that both specify the most significant bits and provide a range of prefix lengths that can match. The difference is that upto allows you to specify an upper limit only for the prefix length range, while prefix-length-range allows you to specify both lower and upper limits.

For more examples of these route list match types, see “Examples: Configure Route Lists” on page 110.

Table 19: Route List Match Types

Match Type	Match If ...
exact	The route shares the same most-significant bits (described by *prefix-length*), and *prefix-length* is equal to the route’s prefix length.
longer	The route shares the same most-significant bits (described by *prefix-length*), and *prefix-length* is greater than the route’s prefix length.
orlonger	The route shares the same most-significant bits (described by *prefix-length*), and *prefix-length* is equal to or greater than the route’s prefix length.
prefix-length-range *prefix-length2–prefix-length3*	The route shares the same most-significant bits (described by *prefix-length*), and the route’s prefix length falls between *prefix-length2* and *prefix-length3*, inclusive.
through *destination-prefix*	All the following are true: ■ The route shares the same most-significant bits (described by *prefix-length*) of the first destination prefix. ■ The route shares the same most-significant bits (described by *prefix-length*) of the second destination prefix for the number of bits in the prefix length. ■ The number of bits in the route’s prefix length is less than or equal to the number of bits in the second prefix. You do not use the through match type in most routing policy configurations. (For an example, see “Examples: Configure Route Lists” on page 110.)
upto *prefix-length2*	The route shares the same most-significant bits (described by *prefix-length*) and the route’s prefix length falls between *prefix-length* and *prefix-length2*.

Table 20: Match Type Examples

Prefix	192.168/16 exact	192.168/16 longer	192.168/16 orlonger	192.168/16 upto /24	192.168/16 through 192.168.16/20	192.168/16 prefix-length-range /18–/20
192.0.0.0/8						
192.168.0.0/16	Match		Match	Match	Match	
192.168.0.0/17		Match	Match	Match	Match	
192.168.0.0/18		Match	Match	Match	Match	Match
192.168.0.0/19		Match	Match	Match	Match	Match
192.168.4.0/24		Match	Match	Match		
192.168.5.4/30		Match	Match			
192.168.12.4/30		Match	Match			
192.168.12.128/32		Match	Match			
192.168.16.0/20		Match	Match	Match	Match	Match
192.168.192.0/18		Match	Match	Match		Match
192.168.224.0/19		Match	Match	Match		Match
192.169.1.0/24						
192.170.0.0/16						

How a Route List Is Evaluated

During route list evaluation, the policy framework software compares each route's source address with the destination prefixes in the route list. The evaluation occurs in two steps:

1. The policy framework software performs a *longest-match lookup*, which means that the software searches for the prefix in the list with the longest length.

   The longest-match lookup considers the prefix and prefix length only and not the match type. The following sample route list illustrates this point:

   ```
   from {
       route-filter 192.168.0.0/14 upto /24 reject;
       route-filter 192.168.0.0/15 exact;
   }
   then accept;
   ```

   The longest match is the second route-filter, 192.168.0.0/15, which is based on prefix and prefix length only.

2. Once an incoming route matches a prefix (longest first), the following occurs:

   - The route filter stops evaluating other prefixes, even if the match type fails.
   - The software examines the match type and action associated with that prefix.

In Step 1, if route 192.168.1.0/24 were evaluated, it would fail to match. It matches the longest prefix of 192.168.0.0/15, but it does not match exact. The route filter is finished because it matched a prefix, but the result is a failed match because the match type failed.

If a match occurs, the action specified with the prefix is taken. If an action is not specified with the prefix, the action in the then statement is taken. If neither action is specified, the software evaluates the next term or routing policy, if present, or takes the accept or reject action specified by the default policy. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 19.

If you specify multiple prefixes in the route list, only one prefix needs to match for a match to occur. The route list matching is effectively a logical OR operation.

If a match does not occur, the software evaluates the next term or routing policy, if present, or takes the accept or reject action specified by the default policy.

For example, compare the prefix 192.168.254.0/24 against the following route list:

```
route-filter 192.168.0.0/16 orlonger;
route-filter 192.168.254.0/23 exact;
```

The prefix 192.168.254.0/23 is determined to be the longest prefix. When evaluating 192.168.254.0/24 against the longest prefix, a match occurs (192.168.254.0/24 is a subset of 192.168.254.0/23). Because of the match between 192.168.254.0/24 and the longest prefix, the evaluation continues. However, when evaluating the match type, a match does not occur between 192.168.254.0/24 and 192.168.254.0/23 exact. The software concludes that the term does not match and goes on to the next term or routing policy, if present, or takes the accept or reject action specified by the default policy.

How Prefix Order Affects Route List Evaluation

The order in which the prefixes are specified (from top to bottom) typically does not matter, because the policy framework software scans the route list looking for the longest prefix during evaluation. An exception to this rule is when you use the same destination prefix multiple times in a list. In this case, the order of the prefixes is important, because the list of identical prefixes is scanned from top to bottom, and the first match type that matches the route applies.

In the following example, different match types are specified for the same prefix. The route 0.0.0.0/0 would be rejected, the route 0.0.0.0/8 would be marked with next-hop self, and the route 0.0.0.0/25 would be rejected.

```
route-filter 0.0.0.0/0 upto /7 reject;
route-filter 0.0.0.0/0 upto /24 next-hop self;
route-filter 0.0.0.0/0 orlonger reject;
```

Common Configuration Problem with the Longest-Match Lookup

A common problem when defining a route list is including a shorter prefix that you want to match with a longer, similar prefix in the same list. For example, imagine that the prefix 192.168.254.0/24 is compared against the following route list:

```
route-filter 192.168.0.0/16 orlonger;
route-filter 192.168.254.0/23 exact;
```

Because the policy-framework software performs longest-match lookup, the prefix 192.168.254.0/23 is determined to be the longest prefix. An exact match does not occur between 192.168.254.0/24 and 192.168.254.0/23 exact. The software concludes that the term does not match and goes on to the next term or routing policy, if present, or takes the accept or reject action specified by the default policy. (For more information about the default routing policies, see "Default Routing Policies and Actions" on page 19.) The shorter prefix 192.168.0.0/16 orlonger that you wanted to match is inadvertently ignored.

One solution to this problem is to remove the prefix 192.168.0.0/16 orlonger from the route list in this term and move it to a previous term where it is the only prefix or the longest prefix in the list.

Examples: Configure Route Lists

The examples in this section show only fragments of routing policies. Normally, you would combine these fragments with other terms or routing policies.

In all examples, remember that the following actions apply to nonmatching routes:

- Evaluate next term, if present.
- Evaluate next policy, if present.
- Take the accept or reject action specified by the default policy. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 19.

Reject routes with a destination prefix of 0.0.0.0 and a mask length from 0 through 8, and accept all other routes:

```
[edit]
policy-options {
    policy-statement from-hall2 {
        term 1 {
            from {
                route-filter 0.0.0.0/0 upto /8 reject;
            }
        }
        then accept;
    }
}
```

Reject routes with a mask of /8 and greater (that is, /8, /9, /10, and so on) and accept routes with the first eight bits set to 0 and at least eight bits in length:

```
[edit]
policy-options {
    policy-statement from-hall3 {
        term term1 {
            from {
                route-filter 0/0 upto /7 accept;
                route-filter 0/8 orlonger;
            }
            then reject;
        }
    }
}
```

Reject routes with the destination prefix of 192.168.10/24 and a mask between /26 and /29 and accept all other routes:

```
[edit]
policy-options {
    policy-statement from-customer-a {
        term term1 {
            from {
                route-filter 192.168.10/24 prefix-length-range /26–/29 reject;
                route-filter 0/0;
            }
            then accept;
        }
    }
}
```

Reject a range of routes from USC hosts, and accept all other routes:

```
[edit]
policy-options {
    policy-statement usc-hosts-only {
        from {
            route-filter 128.125.0.0/16 upto /31 reject;
            route-filter 0/0;
        }
        then accept;
    }
}
```

You do not use the through match type in most routing policy configurations. You should think of through as a tool to group a contiguous set of exact matches. For example, instead of specifying four exact matches:

```
from route-filter 0.0.0.0/1 exact
from route-filter 0.0.0.0/2 exact
from route-filter 0.0.0.0/3 exact
from route-filter 0.0.0.0/4 exact
```

You could represent them with the following single match:

```
from route-filter 0.0.0.0/1 through 0.0.0.0/4
```

Explicitly accept a limited set of prefixes (in the first term) and reject all others (in the second term):

```
[edit policy-options]
policy-statement internet-in {
    term 1 {
        from {
            route-filter 192.168.231.0/24 exact accept;
            route-filter 192.168.244.0/24 exact accept;
            route-filter 192.200.198.0/24 exact accept;
            route-filter 192.200.160.0/24 exact accept;
            route-filter 192.200.59.0/24 exact accept;
        }
    }
    term 2 {
        then {
            reject;
        }
    }
}
```

Reject a few groups of prefixes, then accept the remaining prefixes:

```
[edit policy-options]
policy-statement drop-routes {
    term 1{
        from {                                  # first, reject a number of prefixes:
            route-filter default exact reject;  # reject 0.0.0.0/0 exact
            route-filter 0.0.0.0/8 orlonger reject;  # reject prefix 0, mask /8 or longer
            route-filter 127.0.0.0/8 orlonger reject;  # reject loopback addresses
            route-filter 128.105.0.0/16 exact {  # accept 128.105.0.0/16 exact
                as-path-prepend "1 2 3";
                accept;
            }
            route-filter 192.0.2.0/24 orlonger reject;  # reject test network packets
            route-filter 224.0.0.0/3 orlonger reject;  # reject multicast and higher
            route-filter 0.0.0.0/0 upto /24 accept;  # accept everything up to /24
            route-filter 0.0.0.0/0 orlonger accept;  # accept everything else
            }
        }
    }
}
```

Reject all prefixes longer than 24 bits. You would install this routing policy in a sequence of routing policies in an export statement. The first term in this filter passes on all routes with a prefix length of up to 24 bits. The second, unnamed term rejects everything else.

```
[edit policy-options]
policy-statement 24bit-filter {
    term acl20 {
        from {
            route-filter 0.0.0.0/0 upto /24;
        }
        then next policy;
    }
    then reject;
}
```

If, in this example, you were to specify route-filter 0.0.0.0/0 upto /24 accept, matching prefixes would be accepted immediately and the next routing policy in the export statement would never get evaluated.

If you were to include the then reject statement in the term acl20, prefixes greater than 24 bits would never get rejected because the policy framework software, when evaluating the term, would move on to evaluating the next statement before reaching the then reject statement.

Configure a routing policy for rejecting Protocol Independent Multicast (PIM) multicast traffic joins for a source destination prefix from a neighbor:

```
[edit]
policy-options {
    policy-statement join-filter {
        from {
            neighbor 10.14.12.20;
            source-address-filter 128.83.0.0/16 orlonger;
        }
        then reject;
    }
}
```

Configure a routing policy for rejecting PIM traffic for a source destination prefix from an interface:

```
[edit]
policy-options {
    policy-statement join-filter {
        from {
            interface so-1/0/0.0;
            source-address-filter 128.83.0.0/16 orlonger;
        }
        then reject;
    }
}
```

The following routing policy qualifiers apply to PIM:

- interface—Interface over which a join is received
- neighbor—Source from which a join originates
- route-filter—Group address
- source-address-filter—Source address for which to reject a join

For more information about importing a PIM join filter in a PIM protocol definition, see the *JUNOS Internet Software Configuration Guide: Multicast*.

Configure Subroutines

You can use a routing policy called from another routing policy as a match condition. This process makes the called policy a *subroutine.*

This section describes the following task for configuring subroutines and provides the following example:

- Define Subroutines on page 114
- Example: Configure a Subroutine on page 118

Define Subroutines

To configure a subroutine in a routing policy to be called from another routing policy, create the subroutine and specify its name using the policy match condition (described in Table 9 on page 41) in the from or to statement of another routing policy:

```
[edit]
policy-options {
    policy-statement subroutine-policy-name {
        term term-name {
            from {
                match-conditions;
                route-filter destination-prefix match-type <actions>;
                source-address-filter destination-prefix match-type <actions>;
                prefix-list name;
            }
            to {
                match-conditions;
            }
            then actions;
        }
    }
}
```

```
policy-options {
   policy-statement policy-name {
      term term-name {
         from {
            policy subroutine-policy-name;
         }
         to {
            policy subroutine-policy-name;
         }
         then actions;
      }
   }
}
```

Do not evaluate a routing policy within itself. If you attempt to do so, no prefixes will ever match the routing policy.

The action specified in a subroutine is used to provide a match condition to the calling policy. If the subroutine specifies an action of accept, the calling policy considers the route to be a match. If the subroutine specifies an action of reject, the calling policy considers the route not to match. If the subroutine specifies an action that is meant to manipulate the route characteristics, the changes are made. For more details about the subroutine evaluation, see “How a Routing Policy Subroutine Is Evaluated” on page 30.

Termination Actions

A subroutine with particular statements can behave differently from a routing policy that contains the same statements. With a subroutine, you must remember that the possible termination actions of accept or reject specified by the subroutine or the default policy can greatly affect the expected results. (For more information about the default routing policies, see “Default Routing Policies and Actions” on page 19.)

In particular, you must consider what happens if a match does not occur with routes specified in a subroutine and if the default policy action that is taken is the action that you expect and want.

For example, imagine that you are a network administrator at an Internet service provider (ISP) that provides service to Customer A. You have configured several routing policies for the different classes of neighbors that Customer A presents on various links. To save time maintaining the routing policies for Customer A, you have configured a subroutine that identifies their routes and various routing policies that call the subroutine, as shown below:

```
[edit]
policy-options {
   policy-statement customer-a-subroutine {
      from {
         route-filter 10.1/16 exact;
         route-filter 10.5/16 exact;
         route-filter 192.168.10/24 exact;
      }
      then accept;
   }
}
```

```
policy-options {
    policy-statement send-customer-a-default {
        from {
            policy customer-a-subroutine;
        }
        then {
            set metric 500;
            accept;
        }
    }
}
policy-options {
    policy-statement send-customer-a-primary {
        from {
            policy customer-a-subroutine;
        }
        then {
            set metric 100;
            accept;
        }
    }
}
policy-options {
    policy-statement send-customer-a-secondary {
        from {
            policy customer-a-subroutine;
        }
        then {
            set metric 200;
            accept;
        }
    }
}
protocols {
    bgp {
        group customer-a {
            export send-customer-a-default;
            neighbor 1.1.1.1;
            neighbor 1.1.2.1;
            neighbor 1.1.3.1 {
                export send-customer-a-primary;
            }
            neighbor 1.1.4.1 {
                export send-customer-a-secondary;
            }
        }
    }
}
```

The following results occur with this configuration:

- The group-level export statement resets the metric to 500 when advertising all BGP routes to neighbors 1.1.1.1 and 1.1.2.1 rather than just the routes that match the subroutine route filters.

- The neighbor-level export statements reset the metric to 100 and 200 when advertising all BGP routes to neighbors 1.1.3.1 and 1.1.4.1, respectively, rather than just the BGP routes that match the subroutine route filters.

These unexpected results occur because the subroutine policy does not specify a termination action for routes that do not match the route filter and therefore, the default BGP export policy of accepting all BGP routes is taken.

If the statements included in this particular subroutine had been contained within the calling policies themselves, only the desired routes would have their metrics reset.

This example illustrates the differences between routing policies and subroutines and the importance of the termination action in a subroutine. Here, the default BGP export policy action for the subroutine was not carefully considered. A solution to this particular example is to add one more term to the subroutine that rejects all other routes that do not match the route filters:

```
[edit]
policy-options {
    policy-statement customer-a-subroutine {
        term accept-exact {
            from {
                route-filter 10.1/16 exact;
                route-filter 10.5/16 exact;
                route-filter 192.168.10/24 exact;
            }
            then accept;

        term reject-others {
            then reject;
        }
    }
}
```

Termination action strategies for subroutines in general include the following:

- Depend upon the default policy action to handle all other routes.
- Add a term that accepts all other routes. (Also see “Side Effects of Omitting the “from” Statement from an Export Policy” on page 58.)
- Add a term that rejects all other routes.

The option that you choose depends upon what you want to achieve with your subroutine. Plan your subroutines carefully.

Example: Configure a Subroutine

Create the subroutine is-customer and call it from the routing policies export-customer and import-customer. In import-customer, the action is taken only on routes that match the route filters defined in is-customer.

```
[edit]
policy-options {
    policy-statement is-customer {
        term match-customer {
            from {
                route-filter 172.100.1.0/24 exact;
                route-filter 171.186.100.0/24 exact;
            }
            then accept;
        }
        term drop-others {
            then reject;
        }
    }

    policy-statement export-customer {
        from policy is-customer;
        then accept;
    }

    policy-statement import-customer {
        from {
            protocol bgp;
            policy is-customer;
        }
        then {
            local-preference 10;
            accept;
        }
    }
}
```

Chapter 6

Configure Extended Actions

This chapter describes how to configure extended actions for a routing policy. These extended actions include criteria that manipulate the route characteristics and are more complex than standard actions. You should have an in-depth understanding of extended actions before configuring them. For a complete list of the actions that manipulate route characteristics, see Table 11 on page 45.

This chapter provides information about configuring the following routing policy actions:

Configure AS Path Prepend Action

You can *prepend* or add one or more autonomous system (AS) numbers at the beginning of an AS path. The AS numbers are added after the local AS number has been added to the path. Prepending an AS path makes a shorter AS path look longer and therefore less preferable to the Border Gateway Protocol (BGP).

For example, from AS 1, there are two equal paths (through AS 2 and AS 3) to reach AS 4. You might want packets from certain sources to use the path through AS 2. Therefore, you must make the path through AS 3 look less preferable so that BGP chooses the path through AS 2. In AS 1, you can prepend multiple AS numbers:

```
[edit]
policy-options {
    policy-statement as-path-prepend {
        term prepend {
            from {
                route-filter 192.168.0.0/16 orlonger;
                route-filter 172.16.0.0/12 orlonger;
                route-filter 10.0.0.0/8 orlonger;
            }
            then as-path-prepend "1 1 1 1";
        }
    }
}
```

Configure AS Path Expand Action

You can *expand* or add one or more AS numbers to an AS sequence. The AS numbers are added before the local AS number has been added to the path. Expanding an AS path makes a shorter AS path look longer and therefore less preferable to BGP. The last AS number in the existing path is extracted and prepended *n* times, where *n* is a number from 1 through 32. This is similar to the AS path prepend action, except that the AS path expand action adds an arbitrary sequence of AS numbers.

For example, from AS 1, there are two equal paths (through AS 2 and AS 3) to reach AS 4. You might want packets from certain sources to use the path through AS 2. Therefore, you must make the path through AS 3 look less preferable so that BGP chooses the path through AS 2. In AS 1, you can expand multiple AS numbers.

```
[edit]
policy-options {
    policy-statement as-path-expand {
        term expand {
            from {
                route-filter 192.168.0.0/16 orlonger;
                route-filter 172.16.0.0/12 orlonger;
                route-filter 10.0.0.0/8 orlonger;
            }
            then as-path-expand last-as count 4;
        }
    }
}
```

For routes from AS 2, this makes the route look like `1 2 2 2 2 2` when advertised, where `1` is from AS 1, the `2` from AS 2 is prepended 4 times, and the final `2` is the original `2` received from the neighbor router.

Configure Class Action

For information about class of service (CoS), see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

Configure Damping Action

BGP *route flapping* describes the situation in which BGP systems send an excessive number of update messages to advertise network reachability information. BGP *flap damping* is a way to reduce the number of update messages sent between BGP peers, thereby reducing the load on these peers without adversely affecting the route convergence time.

Flap damping reduces the number of update messages by marking routes as ineligible for selection as the active or preferable route. Doing this leads to some delay, or *suppression*, in the propagation of route information, but the result is increased network stability. You typically apply flap damping to external BGP (EBGP) routes (that is, to routes in different ASs). You can also apply it within a confederation, between confederation member ASs. Because routing consistency within an AS is important, do not apply flap damping to internal BGP (IBGP) routes. (If you do, it is ignored.)

BGP flap damping is defined in RFC 2439, *BGP Route Flap Damping.*

To effect changes to the default BGP flap damping values, you define actions by creating a named set of damping parameters and including it in a routing policy with the damping action (described in Table 11 on page 45). For the damping routing policy to work, you also must enable BGP route flap damping.

This section describes the following:

Configure Flap Damping Parameters

To define damping parameters, include the damping statement at the [edit policy-options] hierarchy level:

```
[edit policy-options]
damping name {
    disable;
    half-life minutes;
    max-suppress minutes;
    reuse number;
    suppress number;
}
```

The name identifies the group of damping parameters. It can contain letters, numbers, and hyphens (–) and can be up to 255 characters. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

You can specify one or more of the damping parameters described in Table 21.

Table 21: Damping Parameters

Damping Parameter	Description	Default	Possible Values
half-life *minutes*	Decay half-life, in minutes	15 minutes	1 through 45 minutes
max-suppress *minutes*	Maximum hold-down time, in minutes	60 minutes	1 through 720 minutes
reuse *minutes*	Reuse threshold	750 (unitless)	1 through 20,000 (unitless)
suppress *minutes*	Cutoff (suppression) threshold	3000 (unitless)	1 through 20,000 (unitless)

If you do not specify one or more of the damping parameters, the default value of the parameter is used.

To understand how to configure these parameters, you need to understand how damping suppresses routes. How long a route can be suppressed is based on a *figure of merit*, which is a value that correlates to the probability of future instability of a route. Routes with higher figure-of-merit values are suppressed for longer periods of time. The figure-of-merit value decays exponentially over time.

A figure-of-merit value of zero is assigned to each new route. The value is increased each time the route is withdrawn or readvertised, or when one of its path attributes changes. With each incident of instability, the value increases as follows:

- Route is withdrawn—1000
- Route is readvertised—1000
- Route's path attributes change—500

When a route's figure-of-merit value reaches a particular level, called the *cutoff* or *suppression threshold*, the route is suppressed. If a route is suppressed, the routing table no longer installs the route into the forwarding table and no longer exports this route to any of the routing protocols. By default, a route is suppressed when its figure-of-merit value reaches 3000. To modify this default, include the suppress option at the [edit policy-options damping *name*] hierarchy level.

If a route has flapped, but then becomes stable so that none of the incidents listed above occur within a configurable amount of time, the figure-of-merit value for the route decays exponentially. The default half-life is 15 minutes. For example, for a route with a figure-of-merit value of 1500, if no incidents occur, its figure-of-merit value is reduced to 750 after 15 minutes and to 375 after another 15 minutes. To modify the default half-life, include the half-life option at the [edit policy-options damping *name*] hierarchy level.

A suppressed route becomes reusable when its figure-of-merit value decays to a value below a *reuse threshold*, thus allowing routes that experience transient instability to once again be considered valid. The default reuse threshold is 750. When the figure-of-merit value passes below the reuse threshold, the route once again is considered usable and can be installed in the forwarding table and exported from the routing table. To modify the default reuse threshold, include the reuse option at the [edit policy-options damping *name*] hierarchy level.

The maximum suppression time provides an upper bound on the time that a route can remain suppressed. The default maximum suppression time is 60 minutes. To modify the default, include the max-suppress option at the [edit policy-options damping *name*] hierarchy level.

A route's figure-of-merit value stops increasing when it reaches a maximum suppression threshold, which is determined based on the route's suppression threshold level, half-life, reuse threshold, and maximum hold-down time.

The merit ceiling, ε_c, which is the maximum merit that a flapping route can collect, is calculated using the following formula:

$$\varepsilon_c \leq \varepsilon_r \, 2^{(t/\lambda)\,(\ln 2)}$$

ε_r is the figure-of-merit reuse threshold, t is the maximum hold-down time in minutes, and λ is the half-life in minutes. For example, if you use the default figure-of-merit values in this formula, but use a half-life of 30 minutes, the calculation is as follows:

$$\varepsilon_c \leq 750 \, 2^{(60/30)\,(\ln 2)}$$
$$\varepsilon_c \leq 3000$$

The cutoff threshold, which you configure using the suppress option, must be less than or equal to the merit ceiling, ε_c. If the configured cutoff threshold or the default cutoff threshold is greater than the merit ceiling, the route is never suppressed and damping never occurs.

To display figure-of-merit information, use the show policy damping command.

A route that has been assigned a figure of merit is considered to have a damping state. To display the current damping information on the router, use the show route detail command.

Define Damping Action

To define the damping action, include the damping action and the name of the configured damping parameters either in a then statement or in a route-filter option in a from statement.

Enable BGP Route Flap Damping

For information about enabling BGP route flap damping, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

Disable Damping by Prefix

Normally, you enable or disable damping on a per-peer basis. However, you can disable damping for a specific prefix received from a peer by including the disable option at the [edit policy-options damping *name*] hierarchy level:

```
[edit policy-options damping name]
disable;
```

Example: Disable by Prefix

In this routing policy example, although damping is enabled for the peer, the damping none statement specifies that damping be disabled for prefix 3.0.0.0/8 in Policy-A. This route is not damped because the routing policy statement named Policy-A filters on the prefix 3.0.0.0/8 and the action points to the damping statement named none. The remaining prefixes are damped using the default parameters.

```
[edit]
policy-options {
    policy-statement Policy-A {
        from {
            route-filter 3.0.0.0/8 exact;
        }
        then damping none;
    }
    damping none {
        disable;
    }
}
```

Example: Configure BGP Flap Damping

Enable BGP flap damping and configure damping parameters:

```
[edit]
routing-options {
    autonomous-system 666;
}
protocols {
    bgp {
        damping;
        group group1 {
            traceoptions {
                file bgp-log size 1m files 10;
                flag damping;
            }
            import damp;
            type external;
            peer-as 10458;
            neighbor 192.168.2.30;
        }
    }
}
policy-options {
    policy-statement damp {
        from {
            route-filter 11.0.0.0/8 exact {
                damping high;
                accept;
            }
            route-filter 15.0.0.0/8 exact {
                damping medium;
                accept;
            }
            route-filter 3.0.0.0/8 exact {
                damping none;
                accept;
            }
        }
    }
    damping high {
        half-life 30;
        suppress 3000;
        reuse 750;
        max-suppress 60;
    }
    damping medium {
        half-life 15;
        suppress 3000;
        reuse 750;
        max-suppress 45;
    }
    damping none {
        disable;
    }
}
```

To display damping parameters for this configuration, use the show policy damping command:

```
user@host> show policy damping
Damping information for "high":
  Halflife: 30 minutes
  Reuse merit: 750 Suppress/cutoff merit: 3000
  Maximum suppress time: 60 minutes
  Computed values:
    Merit ceiling: 3008
    Maximum decay: 24933
Damping information for "medium":
  Halflife: 15 minutes
  Reuse merit: 750 Suppress/cutoff merit: 3000
  Maximum suppress time: 45 minutes
  Computed values:
    Merit ceiling: 6024
    Maximum decay: 12449
Damping information for "none":
    Damping disabled
```

Configure Load-Balance Per-Packet Action

By default, when there are multiple equal-cost paths to the same destination for the active route, the JUNOS software chooses at random one of the next-hop addresses to install into the forwarding table. Whenever the set of next hops for a destination changes in any way, the next-hop address is rechosen, also at random.

You can configure the JUNOS software so that, for the active route, all next-hop addresses for a destination are installed in the forwarding table. This feature is called per-packet load balancing. You can use load balancing to spread traffic across multiple paths between routers. The behavior of the load-balance per-packet function depends on the version of the Internet Protocol ASIC in your router.

On routers with an Internet Processor ASIC, when per-packet load balancing is configured, traffic between routers with multiple paths is spread at random across the available interfaces. The forwarding table balances the traffic headed to a destination, transmitting it in round-robin fashion among the multiple next hops (up to a maximum of eight equal-cost load-balanced paths). The traffic is load-balanced on a per-packet basis.

On routers with the Internet Processor II ASIC, when per-packet load balancing is configured, traffic between routers with multiple paths is divided into individual traffic flows (up to a maximum of 16 equal-cost load-balanced paths). Packets for each individual flow are kept on a single interface.

To configure the load-balance per-packet action, include the load-balance per-packet action in a then statement or a route-filter option in a from statement in a routing policy.

You must apply the routing policy to routes exported from the routing table to the forwarding table to complete the configuration. To do this, include the export statement at the [edit routing-options forwarding-table] hierarchy level (for routing instances, include the statement at the [edit routing-instances *routing-instance-name* routing-options] hierarchy level).

You can also configure additional information about flows, such as source and destination port number and ingress interface information, to further identify them. By default, the software ignores port data when determining flows. To enable per-flow load balancing, you must set the load-balance per-packet action in the routing policy configuration; for more information about this action, see "Configure Routing Policy" on page 37.
To include port data in the flow determination, include the family inet statement at the [edit forwarding-options hash-key] hierarchy level:

```
[edit forwarding-options hash-key]
family inet {
    layer-3;
    layer-4;
}
```

You must include the layer-3 statement. If you omit the layer-3 statement, the management process removes the hash-key statement from the configuration and the router behaves as if you specified layer-3.

If you include both the layer-3 and layer-4 statements, the router uses the following Layer 3 and Layer 4 information to load-balance:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index

The router recognizes packets in which all of these layer-3 and layer-4 parameters are identical, and ensures that these packets are sent out through the same interface. This prevents problems that might otherwise occur with packets arriving at their destination out of their original sequence.

This is appropriate behavior for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets. For Internet Control Message Protocol (ICMP) packets, the field location offset is the checksum field, which makes each ping packet a separate "flow." This can be problematic; for example, some traceroute implementations might use ICMP rather than UDP for the outgoing packets.

By default, or if you specify only the layer-3 statement, the router uses the following Layer 3 information in the packet header to load-balance:

- Source IP address
- Destination IP address
- Protocol

Load Balancing Based on the MPLS Label Information

For aggregated Ethernet and aggregated SONET interfaces only, to load-balance based on the multiprotocol label switching (MPLS) label information, include the family mpls statement at the [edit forwarding-options hash-key] hierarchy level:

```
[edit forwarding-options hash-key]
family mpls {
    label-1;
    label-2;
}
```

Examples: Configure Per-Packet Load Balancing

Perform per-packet load balancing for all routes:

```
[edit]
policy-options {
    policy-statement load-balancing-policy {
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

Perform per-packet load balancing only for a limited set of routes:

```
[edit]
policy-options {
    policy-statement load-balancing-policy {
        from {
            route-filter 192.168.10/24 orlonger;
            route-filter 9.114/16 orlonger;
        }
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

Chapter 7

Summary of Routing Policy Configuration Statements

The following sections explain each of the routing policy configuration statements. The statements are organized alphabetically.

apply-path

Syntax apply-path *path*;

Hierarchy Level [edit policy-options prefix-list *name*]

Description Expand a prefix-list to include all prefixes pointed to by a defined path.

Options *path*—A string of elements composed of identifiers or configuration keywords that points to a set of prefixes. You can include wildcards (enclosed in angle brackets) to match more than one identifier.

prefix-list *name*—Name of a list of Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) prefixes. To create a named list of IP address prefixes, see "Configure Extended Match Conditions" on page 83.

Usage Guidelines See "Configure Prefix Lists" on page 102.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

as-path

Syntax as-path *name regular-expression*;

Hierarchy Level [edit policy-options]

Description Define an autonomous system (AS) path regular expression for use in a routing policy match condition.

Options *name*—Name that identifies the regular expression. The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (double quotes).

regular-expression—One or more regular expressions used to match the AS path.

Usage Guidelines See "Configure AS Path Regular Expressions" on page 83.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

as-path-group

Syntax
```
as-path-group group-name {
    [ as-path name regular-expression ];
}
```

Hierarchy Level [edit policy-options]

Description Define an autonomous system (AS) path regular expression for use in a routing policy match condition.

Options *group-name*—Name that identifies the AS path group. One or more AS path regular expressions must be listed below the as-path-group hierarchy.

name—Name that identifies the regular expression. The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (double quotes).

regular-expression—One or more regular expressions used to match the AS path.

Usage Guidelines See "Configure AS Path Regular Expressions" on page 83.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

community

Syntax

```
community name {
    bandwidth as:bandwidth;
    invert-match;
    members [ community-ids ];
}
```

Hierarchy Level [edit policy-options]

Description Define a community or extended community for use in a routing policy match condition.

Options *name*—Name that identifies the regular expression. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters. To include spaces in the name, enclose it in quotation marks (double quotes).

bandwidth *as*:*bandwidth*—Link bandwidth community attribute. as specifies the autonomous system and bandwidth specifies bandwidth in bytes per second.

invert-match—Invert the results of the community expression matching.

members *community-ids*—One or more community members. If you specify more than one member, you must enclose all members in brackets.

The format for community identifiers is:

as-number:*community-value*

as-number is the AS number and can be a value in the range 1 through 65,534. *community-value* is the community identifier and can be a number in the range 0 through 65,535.

You also can specify *community-ids* for communities as one of the following well-known community names, which are defined in RFC 1997:

- no-export—Routes containing this community name are not advertised outside a Border Gateway Protocol (BGP) confederation boundary.
- no-advertise—Routes containing this community name are not advertised to other BGP peers.
- no-export-subconfed—Routes containing this community name are not advertised to external BGP peers, including peers in other members' ASs inside a BGP confederation.

You can explicitly exclude BGP community information with a static route using the none option. Include none when configuring an individual route in the route portion of the static statement to override a community option specified in the defaults portion of the statement.

The format for extended community identifiers is:

type:*administrator*:*assigned-number*

type is the type of extended community and can be either a target, origin, or domain-id community. The target community identifies the destination to which the route is going. The origin community identifies where the route originated. The domain-id community identifies the Open Shortest Path First (OSPF) domain from which the route originated.

administrator is the administrator. It is either an AS number or an IPv4 address prefix, depending on the type of extended community.

assigned-number identifies the local provider.

Usage Guidelines See “Configure Communities” on page 90.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

damping

Syntax

```
damping name {
    disable;
    half-life minutes;
    max-suppress minutes;
    reuse number;
    suppress number;
}
```

Hierarchy Level [edit policy-options]

Description Define route flap damping properties to set on BGP routes.

Options

disable—Disable damping on a per-prefix basis. Any damping state that is present in the routing table for a prefix is deleted if damping is disabled.

half-life *minutes*—Decay half-life. *minutes* is the interval after which the accumulated figure-of-merit value is reduced by half if the route remains stable.
Range: 1 through 45
Default: 15 minutes

max-suppress *minutes*—Maximum hold-down time. *minutes* is the maximum time that a route can be suppressed no matter how unstable it has been.
Range: 1 through 720
Default: 60 minutes

name—Name that identifies the set of damping parameters. The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (double quotes).

reuse *number*—Reuse threshold. *number* is the figure-of-merit value below which a suppressed route can be used again.
Range: 1 through 20,000
Default: 750 (unitless)

suppress *number*—Cutoff (suppression) threshold. *number* is the figure-of-merit value above which a route is suppressed for use or inclusion in advertisements.
Range: 1 through 20,000
Default: 3000 (unitless)

Usage Guidelines See “Configure Flap Damping Parameters” on page 121.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

export

Syntax export [*policy-names*];

Hierarchy Level [edit protocols *protocol-name*]

Description Apply one or more policies to routes being exported from the routing table into a routing protocol.

Options *policy-names*—Names of one or more policies defined with a policy-statement statement.

Usage Guidelines See “Apply a Routing Policy” on page 52.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

import

Syntax import [*policy-names*];

Hierarchy Level [edit protocols *protocol-name*]

Description Apply one or more policies to routes being imported into the routing table from a routing protocol.

Options *policy-names*—Names of one or more policies defined with a policy-statement statement.

Usage Guidelines See “Apply a Routing Policy” on page 52.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

policy-options

Syntax policy-options { ... }

Hierarchy Level [edit]

Description Configure routing policy.

Options The statements are explained separately.

Usage Guidelines See “Define Routing Policies” on page 37.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

policy-statement

Syntax

```
policy-statement policy-name {
    term term-name {
        from {
            family family-name;
            match-conditions;
            policy subroutine-policy-name;
            prefix-list name;
            route-filter destination-prefix match-type <actions>;
            source-address-filter destination-prefix match-type <actions>;
        }
        to {
            match-conditions;
            policy subroutine-policy-name;
        }
        then actions;
    }
}
```

Hierarchy Level [edit policy-options]

Description Define a routing policy, including subroutine policies.

Options

actions—(Optional) One or more actions to take if the conditions match. The actions are described in Table 10 on page 44 and Table 11 on page 45.

family *family-name*—(Optional) Specify an address family protocol. Specify inet for an IPv4 address protocol. Specify inet6 for a 128-bit IPv6 address protocol, and to enable interpretation of IPv6 router filter addresses. When family is not specified, the router uses the default IPv4 setting.

from—(Optional) Match a route based on its source address.

match-conditions—(Optional in from statement; required in to statement) One or more conditions to use to make a match. The qualifiers are described in Table 9 on page 41.

policy *subroutine-policy-name*—Use another policy as a match condition within this policy. The name identifying the subroutine policy can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”). For information about how to configure subroutines, see “Configure Subroutines” on page 114.

policy-name—Name that identifies the policy. The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (double quotes).

prefix-list *name*—Name of a list of IPv4 or IPv6 prefixes. To create a named list of IP address prefixes, see “Configure Extended Match Conditions” on page 83.

route-filter *destination-prefix match-type <actions>*—(Optional) List of routes on which to perform an immediate match. *destination-prefix* is the IPv4 or IPv6 route prefix to match, *match-type* is the type of match (see Table 19 on page 107), and *actions* is the action to take if the *destination-prefix* matches.

source-address-filter *destination-prefix match-type <actions>*—(Optional) Multicast source addresses in multiprotocol BGP (MBGP) and Multicast Source Discovery Protocol (MSDP) environments on which to perform an immediate match. *destination-prefix* is the IPv4 or IPv6 route prefix to match, *match-type* is the type of match (see Table 19 on page 107), and *actions* is the action to take if the *destination-prefix* matches.

term *term-name*—Name that identifies the term.

to—(Optional) Match a route based on its destination address or the protocols into which the route is being advertised.

then—(Optional) Actions to take on matching routes. The actions are described in Table 10 on page 44 and Table 11 on page 45.

Usage Guidelines See "Define Routing Policies" on page 37 and "Configure Extended Match Conditions" on page 83.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

prefix-list

Syntax
```
prefix-list name {
    ip-addresses;
}
```

Hierarchy Level [edit policy-options]

Description Define a list of IPv4 or IPv6 address prefixes for use in a routing policy statement or firewall filter statement.

Options *name*—Name that identifies the list of IPv4 or IPv6 address prefixes.

ip-addresses—List of IPv4 or IPv6 address prefixes, one IP address per line in the configuration.

Usage Guidelines See "Configure Prefix Lists" on page 102.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Part 3

Firewall Filters

Chapter 8
Firewall Filter Overview

Firewall filters allow you to filter packets based on their components and to perform an action on packets that match the filter.


Note

The JUNOS Internet software provides a *policy framework*, which is a collection of JUNOS policies that include routing policies and firewall filter policies. These policies share some fundamental similarities. (For information about the similarities and differences among these policies, see "Policy Framework Overview" on page 3.) However, when referring to a firewall filter policy in the firewall filters part of the manual, the term *firewall filter* is used.

Depending on the hardware configuration of the router, you can use firewall filters for the following purposes:

- On routers equipped with an Internet Processor II ASIC, you can control *data packets*, which are chunks of data transiting the router as they are forwarded from a source to a destination.
- On all routers, you can control the *local packets*, which are chunks of data that are destined for or sent by the Routing Engine.

With the Internet Processor II ASIC, you can use filters on data packets passing through the router to provide protocol-based firewalls, thwart denial-of-service (DoS) attacks, prevent falsifying of source addresses, create access control lists, and implement rate limiting (policing). (Use the show chassis hardware command to determine whether a router has an Internet Processor or an Internet Processor II ASIC.)

You can use the filters to restrict the local packets that pass from the router's physical interfaces to the Routing Engine. Such filters are useful in protecting the IP services that run on the Routing Engine, such as telnet, secure shell (ssh), and the Border Gateway Protocol (BGP), from denial-of-service attacks. You can define input filters, which affect only inbound traffic destined for the Routing Engine, and output filters, which affect only outbound traffic sent from the Routing Engine. You can also use policing, or rate limiting, to provide a finer level of control over local packets destined for the Routing Engine.


Note

In the remainder of the firewall filters part of this manual, the term *packets* refers to both data and local packets unless explicitly stated otherwise.

You can apply firewall filters to packets entering or leaving the router on one, more than one, or all interfaces. For each interface, you can apply a firewall filter to incoming or outgoing traffic, or both, and the same filter can be used for both.

You can define firewall filters that apply to Internet Protocol Version 4 (IPv4), Internet Protocol Version 6 (IPv6), or Multiprotocol Label Switching (MPLS) traffic.

There is no limit to the number of filters and counters you can set, but there are some practical considerations. More counters require more terms, and a large number of terms can take a long time to process during a commit. However, filters with more than 1000 terms and counters have been implemented successfully.

Firewall Filter Components

In a firewall filter, you first define the address structure type (IPv4, IPv6, or MPLS), then you define one or more terms that specify the filtering criteria and the action to take if a match occurs. Each term consists of two components:

- Match conditions—Values or fields that the packet must contain. You can define various match conditions, including the IP source address field, IP destination address field, Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) source port field, IP protocol field, Internet Control Message Protocol (ICMP) packet type, IP options, TCP flags, incoming logical or physical interface, and outgoing logical or physical interface.

- Action—Specifies what to do if a packet matches the match conditions. Possible actions are to accept, discard, or reject a packet, go to the next term, or take no action. In addition, statistical information can be recorded for a packet: it can be counted, logged, or sampled.

The order of the terms within a firewall filter is significant. Packets are tested against each term in the order in which they are listed in the configuration. When the first matching conditions are found, the action associated with that term is applied to the packet and the evaluation of the firewall filter ends, unless the next term action modifier is included. If the next term action is included, the matching packet is then evaluated against the next term in the firewall filter; otherwise, the matching packet is not evaluated against subsequent terms in the firewall filter.

If, after all terms are evaluated, a packet matches no terms in a filter, the packet is silently discarded.

If a packet arrives on an interface and a firewall filter is not configured for the incoming traffic on that interface, the packet is accepted by default.

Although policing, traffic sampling, and forwarding are configured as firewall filters, they are documented in separate parts of this manual. For information about policing, see “Policer Configuration” on page 185. For information about traffic sampling and forwarding, see “Traffic Sampling and Forwarding Configuration” on page 213.

Chapter 9
Firewall Filter Configuration

To configure firewall filters, you include statements at the [edit] hierarchy level of the configuration:

```
[edit]
firewall {
    family family-name {
        filter filter-name {
            accounting-profile name
            interface-specific
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    action;
                    action-modifiers;
                }
            }
        }
    }
}
```

The JUNOS software supports the following Requests for Comments (RFCs) related to filtering:

- RFC 792, *Internet Control Message Protocol (ICMP)*
- RFC 2474, *Definition of the Differentiated Services (DS) Field*
- RFC 2475, *An Architecture for Differentiated Services*
- RFC 2597, *Assured Forwarding PHB*
- RFC 2598, *An Expedited Forwarding PHB*
- RFC 2460, *Internet Protocol Version 6*

This chapter describes the following tasks for configuring firewall filters:

Minimum Firewall Filter Configuration

To configure a firewall filter, you must perform at least the following tasks:

- Configure firewall filters—To configure firewall filters, include the family *family-name* statement and one or more filter statements at the [edit firewall] hierarchy level:

```
[edit]
firewall {
    family family-name {
        filter filter-name {
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    action;
                    action-modifiers;
                }
            }
        }
    }
}
```

- Apply firewall filters to interfaces—Firewall filters control local packets to and from the Routing Engine if they are applied to the loopback interface, lo0. With the Internet Processor II ASIC, firewall filters can control data packets through the router when they are applied to an external interface. To have a firewall filter take effect, you must apply it to an interface by including the filter statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```
[edit]
interfaces {
    interface-name {
        unit logical-unit-number {
            family family-name {
                filter {
                    input filter-name;
                    output filter-name;
                }
            }
        }
    }
}
```

Configure Firewall Filters

To configure firewall filters, include the firewall statement at the [edit] hierarchy level:

```
[edit]
firewall {
    family family-name {
        filter filter-name {
            accounting-profile name
            interface-specific
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    action;
                    action-modifiers;
                }
            }
        }
    }
}
```

You can specify IPv4 filters at either the [edit firewall] hierarchy level or the [edit firewall family inet] hierarchy level.

The following sections describe the components of the firewall statement and provide examples of configuring firewall filters:

- Configure the Family Address Type on page 144
- Configure the Filter Name on page 144
- Configure the Filter Terms on page 144
- Configure a Filter Match Statement on page 145
- Configure a Filter Action Statement on page 145
- How Firewall Filters Are Evaluated on page 149

- Filter Match Conditions on page 150
- How Firewall Filters Test a Packet's Protocol on page 161
- Examples: Define Firewall Filters on page 163

Configure the Family Address Type

To configure the family address type for a firewall filter, include the family statement at the [edit firewall] hierarchy level:

```
[edit firewall]
family family-name { ... }
```

Specify the address family, either IPv4 (inet), IPv6 (inet6), or MPLS.

Configure the Filter Name

To configure the filter name, include the filter statement:

```
filter filter-name { ... }
```

For IPv4 traffic, configure the filter name at the [edit firewall family inet] hierarchy level. For IPv6 traffic, configure the filter name at the [edit firewall family inet6] hierarchy level. The filter name can contain letters, numbers, and hyphens (–) and can be up to 24 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

Configure the Filter Terms

Each firewall filter consists of one or more *terms*. To configure a term, include the term statement:

```
term term-name { ... }
```

For IPv4 traffic, configure the filter terms at the [edit firewall family inet filter *filter-name*] hierarchy level. For IPv6 traffic, configure the filter terms at the [edit firewall family inet6 filter *filter-name*] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name*] hierarchy level.

The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

Each term name must be unique within a filter.

You can specify multiple terms in a filter, effectively chaining together a series of match–action operations to apply to the packets on an interface. You can also use the next term action so that, when a match condition is met, the evaluation continues to the next term, rather than terminating.

Firewall filter terms are evaluated in the order in which you specify them in the configuration. To reorder terms, use the configuration mode insert command. For example, the command insert term up before term start places the term up before the term start.

Configure a Filter Match Statement

In a firewall filter term, you can define conditions used to match the components of a packet. To configure match conditions, include the from statement:

```
from {
    match-conditions;
}
```

For IPv4 traffic, configure the match conditions at the [edit firewall family inet filter *filter-name* term *term-name*] hierarchy level. For IPv6 traffic, configure the match conditions at the [edit firewall family inet6 filter *filter-name* term *term-name*] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name*] hierarchy level.

You can specify zero or more match conditions in a single from statement. For a match to occur, the packet must match all the conditions in the term. For more information about match conditions, see “Filter Match Conditions” on page 150.

The from statement is optional. If you omit it, all packets are considered to match.

Configure a Filter Action Statement

In a firewall filter term, you can specify the action to take if the packet matches the conditions you have configured in the term. To configure a filter action, include the then statement:

```
then {
    action;
    action-modifiers;
}
```

For IPv4 traffic, configure the filter action at the [edit firewall family inet filter *filter-name* term *term-name*] hierarchy level. For IPv6 traffic, configure the filter action at the [edit firewall family inet6 filter *filter-name* term *term-name*] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name*] hierarchy level.

You can specify zero or one then statement in a filter term. If you omit the then statement or do not specify an action, the packets that match the conditions in the from statement are accepted.

We strongly recommend that you always explicitly configure an action in the then statement.

You can specify one of the following filter actions:

- accept—The packet is accepted and is sent to its destination.
- discard—The packet is not accepted and is not processed further. Discarded packets cannot be logged or sampled.
- next term—Evaluate the next term in the firewall filter.

- reject—The packet is not accepted and a rejection message is returned. Rejected packets can be logged or sampled.
- routing-instance—The packet is accepted and routed by the specified routing instance. For more information, see “Configure Filter-Based Forwarding” on page 174.

In the filter action statement, you can also specify one or more of the following action modifiers:

- count—Add packet to a count total.
- forwarding-class—Specify the packet forwarding class name.
- log—Store the packet’s header information on the Routing Engine.
- loss-priority—Set the packet loss priority (PLP) to low or high.
- policer—Apply rate-limiting procedures to the traffic. For more information, see “Policer Configuration” on page 185.
- sample—Sample the packet traffic. Apply this option only if you have enabled traffic sampling. For more information, see “Traffic Sampling and Forwarding Configuration” on page 213.
- syslog—Log an alert for the packet.
- ipsec-sa *sa-name*—Specify an IPSec security association for the packet. This is used with the source-address and destination-address match conditions.

You can include zero or one action statement, but any combination of action modifiers. For the action or action modifier to take effect, all conditions in the from statement must match. If you specify log as one of the actions in a term, this constitutes a termination action; whether any additional terms in the filter are processed depends on the traffic through the filter.

The action modifier operations carry a default accept action. For example, if you specify an action modifier and do not specify an action, the specified action modifier is implemented and the packet is accepted.

Policing uses a specific type of action, known as a policer action. For more information, see “Policer Configuration” on page 185.

For more information about forwarding classes and loss priority, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

Table 22 describes the filter actions and action modifiers.

Table 22: Firewall Filter Actions and Action Modifiers

Action or Action Modifier	Description
Actions	
accept	Accept a packet. This is the default.
discard	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are not available for logging or sampling.
next term	Continue to the next term for evaluation.
reject <*message-type*>	Discard a packet, sending an ICMP destination unreachable message. Rejected packets can be logged or sampled if you configure either of those action modifiers. You can specify one of the following message codes: administratively-prohibited (default), bad-host-tos, bad-network-tos, host-prohibited, host-unknown, host-unreachable, network-prohibited, network-unknown, network-unreachable, port-unreachable, precedence-cutoff, precedence-violation, protocol-unreachable, source-host-isolated, source-route-failed, or tcp-reset. If you specify tcp-reset, a Transmission Control Protocol (TCP) reset is returned if the packet is a TCP packet. Otherwise, nothing is returned.
routing-instance *routing-instance*	Specify a routing instance to which packets are forwarded.
Action Modifiers	
count *counter-name*	Increment a counter for this filter. The name can contain letters, numbers, and hyphens (–), and can be up to 24 characters long. A counter name is specific to the filter that uses it, so all interfaces that use the same filter count into the same counter.
forwarding-class *class-name*	Specify a particular forwarding class.
ipsec-sa *sa-name*	Specify an IPSec security association for the packet. Used with the source-address and destination-address match conditions.
log	Log the packet's header information in the Routing Engine. You can access this information by issuing the show log command at the command-line interface (CLI).
loss-priority *priority*	Set the PLP to low or high.
policer *policer-name*	Apply rate limits to the traffic using the named policer.
sample	Sample the traffic on the interface. Use this modifier only when traffic sampling is enabled. For more information, see "Traffic Sampling and Forwarding Configuration" on page 213.
syslog	Log an alert for this packet. The log can be sent to a server for storage and analysis.

Example: Configure a Filter Action Statement

Count, sample, and accept the traffic:

```
term all {
   then {
      count sam-1;
      sample;                                  # default action is accept
   }
}
```

Display the packet counter:

```
user@host> show firewall filter sam

Filter/Counter       Packet count         Byte count
sam
   sam-1             98                   8028
```

Display the firewall log output:

```
user@host> show firewall log

Time     Filter     A Interface        Pro Source address  Destination address
23:09:09 -          A at-2/0/0.301     TCP 10.2.0.25       211.211.211.1:80
23:09:07 -          A at-2/0/0.301     TCP 10.2.0.25       211.211.211.1:56
23:09:07 -          A at-2/0/0.301     ICM 10.2.0.25       211.211.211.1:49552
23:02:27 -          A at-2/0/0.301     TCP 10.2.0.25       211.211.211.1:56
23:02:25 -          A at-2/0/0.301     TCP 10.2.0.25       211.211.211.1:80
23:01:22 -          A at-2/0/0.301     ICM 10.2.2.101      211.211.211.1:23251
23:01:21 -          A at-2/0/0.301     ICM 10.2.2.101      211.211.211.1:16557
23:01:20 -          A at-2/0/0.301     ICM 10.2.2.101      211.211.211.1:29471
23:01:19 -          A at-2/0/0.301     ICM 10.2.2.101      211.211.211.1:26873
```

This output file contains the following fields:

- Time—Time at which the packet was received (not shown in the default).
- Filter—Name of a filter that has been configured with the filter statement at the [edit firewall] hierarchy level. A hyphen (–) or the abbreviation pfe indicates that it was handled by the router's Packet Forwarding Engine. A space (no hyphen) indicates that the packet was handled by the Routing Engine.
- A—Filter action:
  - A—Accept (or next term)
  - D—Discard
  - R—Reject
- Interface—Interface on which the filter is configured.
- Pro—Packet's protocol name or number.
- Source address—Source IP address in the packet.
- Destination address—Destination IP address in the packet.

Display the sampling output:

```
user@host> show log /var/tmp/sam

# Apr  7 15:48:50
Time                    Dest           Src Dest Src Proto TOS Pkt Intf  IP   TCP
                        addr          addr port port          len num frag flags
Apr 7 15:48:54 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:55 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:56 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
```

When you enable reverse path forwarding (RPF) on an interface with an input filter for firewall log and count, the input firewall filter does not log the packets rejected by RPF, although the rejected packets are counted. To log the rejected packets, use an RPF check fail filter to log the rejected packets.

For more information about sampling output, see "Configure a Forwarding Table Filter" on page 217.

How Firewall Filters Are Evaluated

When a firewall filter consists of a single term, the filter is evaluated as follows:

- If the packet matches all the conditions, the action in the then statement is taken.
- If the packet matches all the conditions, and if there is no action specified in the then statement, the default action accept is used.
- If the packet does not match all the conditions, it is discarded.

When a firewall filter consists of more than one term, the filter is evaluated sequentially:

- The packet is evaluated against the conditions in the from statement in the first term.
- If the packet matches, the action in the then statement is taken and, if the next term action is not used, the evaluation ends. Subsequent terms in the filter are not evaluated.
- If the packet matches, the action in the then statement is taken; if the next term action is present, the evaluation continues to the next term.
- If the packet does not match, it is evaluated against the conditions in the from statement in the second term.

  This process continues until either the packet matches the from conditions in one of the subsequent terms or there are no more terms.
- If a packet passes through all the terms in the filter without matching any of them, it is discarded.

If a term does not contain a from statement, the packet is considered to match and the action in the term's then statement is taken.

If a term does not contain a then statement or if you do not configure an action in the then statement, and if the packet matches the conditions in the term's from statement, the packet is accepted.

Each firewall filter has an implicit discard action at the end of the filter, which is equivalent to the following explicit filter term:

```
term implicit-rule {
    then discard;
}
```

Therefore, if a packet matches none of the terms in the filter, it is discarded.

Filter Match Conditions

In the from statement in the firewall filter term, you specify conditions that the packet must match for the action in the then statement to be taken. All conditions in the from statement must match for the action to be taken. The order in which you specify match conditions is not important, because a packet must match all the conditions in a term for a match to occur.

If you specify no match conditions in a term, that term matches all packets.

An individual condition in a from statement can contain a list of values. For example, you can specify numeric ranges or multiple source or destination addresses. When a condition defines a list of values, a match occurs if one of the values in the list matches the packet.

Individual conditions in a from statement can be negated. When you negate a condition, you are defining an explicit mismatch. If a packet matches a negated condition, it is immediately considered not to match the from statement, and the next term in the filter is evaluated, if there is one; if there are no more terms, the packet is discarded.

Match conditions are grouped into the following categories depending upon how you specify the condition:

- Specify Numeric Range Filter Match Conditions on page 151
- Specify Address Filter Match Conditions on page 154
- Specify Multiple Match Conditions on page 157
- Specify Bit-Field Filter Match Conditions (IPv4 Traffic Only) on page 158
- Specify Class-based Filter Match Conditions on page 160

Specify Numeric Range Filter Match Conditions

Numeric range filter conditions match packet fields that can be identified by a numeric value, such as port and protocol numbers. For numeric range filter match conditions, you specify a keyword that identifies the condition and a single value or a range of values that a field in a packet must match. Table 23 describes the numeric range filter match conditions for IPv4 addresses, and Table 24 describes them for IPv6 addresses.

You can specify the numeric range value in one of the following ways:

- Single number. A match occurs if the value of the field matches the number. For example:

  ```
  source-port 25;
  ```

- Range of numbers. A match occurs if the value of the field falls within the specified range. The following example matches source ports 1024 through 65,535 inclusive:

  ```
  source-port 1024-65535;
  ```

- Text synonym for a single number. A match occurs if the value of the field matches the number that corresponds to the synonym. For example:

  ```
  source-port smtp;
  ```

To specify multiple values in a single match condition, group the values within square brackets following the keyword. For example:

```
source-port [smtp ftp-data 25 1024-65535];
```

To exclude a numeric value, append the string -except to the match keyword. For example, the following condition would match only if the source port is not 25:

```
source-port-except 25;
```

The following condition would match only if the port number is not one of those in the list:

```
source-port-except [smtp ftp-data 666 1024-65535];
```

Table 23: Numeric Range IPv4 Firewall Filter Match Conditions

Match Condition	Description
keyword-except	Negate a match. For example, destination-port-except *number*.
ah-spi *spi-value*	IPSec authentication header (AH) security parameter index (SPI) value. Match on this specific SPI value.
ah-spi-except *spi-value*	IPSec AH SPI value. Do not match on this specific SPI value.
destination-port *number*	TCP or UDP destination port field. You cannot specify both the port and destination-port match conditions in the same term. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the port numbers are also listed): afs (1483), bgp (179), biff (512), bootpc (68), bootps (67), cmd (514), cvspserver (2401), dhcp (67), domain (53), eklogin (2105), ekshell (2106), exec (512), finger (79), ftp (21), ftp-data (20), http (80), https (443), ident (113), imap (143), kerberos-sec (88), klogin (543), kpasswd (761), krb-prop (754), krbupdate (760), kshell (544), ldap (389), login (513), mobileip-agent (434), mobilip-mn (435), msdp (639), netbios-dgm (138), netbios-ns (137), netbios-ssn (139), nfsd (2049), nntp (119), ntalk (518), ntp (123), pop3 (110), pptp (1723), printer (515), radacct (1813), radius (1812), rip (520), rkinit (2108), smtp (25), snmp (161), snmptrap (162), snpp (444), socks (1080), ssh (22), sunrpc (111), syslog (514), tacacs-ds (65), talk (517), telnet (23), tftp (69), timed (525), who (513), xdmcp (177), zephyr-clt (2103), or zephyr-hm (2104).
dscp *number*	Differentiated Services code point (DSCP). The DiffServ protocol uses the type of service (ToS) byte in the IP header. The most significant six bits of this byte form the DSCP. For more information, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): The Expedited Forwarding RFC defines one code point: ef (46). The Assured Forwarding RFC defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: af11 (10), af12 (12), af13 (14), af21 (18), af22 (20), af23 (22), af31 (26), af32 (28), af33 (30), af41 (34), af42 (36), af43 (38)
esp-spi *spi-value*	IPSec encapsulating security payload (ESP) SPI value. Match on this specific SPI value.
esp-spi-except *spi-value*	IPSec ESP SPI value. Do not match on this specific SPI value.
fragment-offset *number*	Fragment offset field.
icmp-code *number*	ICMP code field. This value or keyword provides more specific information than icmp-type. Because the value's meaning depends upon the associated icmp-type, you must specify icmp-type along with icmp-code. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated: parameter-problem: ip-header-bad (0), required-option-missing (1) redirect: redirect-for-host (1), redirect-for-network (0), redirect-for-tos-and-host (3), redirect-for-tos-and-net (2) time-exceeded: ttl-eq-zero-during-reassembly (1), ttl-eq-zero-during-transit (0) unreachable: communication-prohibited-by-filtering (13), destination-host-prohibited (10), destination-host-unknown (7), destination-network-prohibited (9), destination-network-unknown (6), fragmentation-needed (4), host-precedence-violation (14), host-unreachable (1), host-unreachable-for-TOS (12), network-unreachable (0), network-unreachable-for-TOS (11), port-unreachable (3), precedence-cutoff-in-effect (15), protocol-unreachable (2), source-host-isolated (8), source-route-failed (5)
icmp-type *number*	ICMP packet type field. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): echo-reply (0), echo-request (8), info-reply (16), info-request (15), mask-request (17), mask-reply (18), parameter-problem (12), redirect (5), router-advertisement (9), router-solicit (10), source-quench (4), time-exceeded (11), timestamp (13), timestamp-reply (14), or unreachable (3).
interface-group *group-number*	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For information about configuration interface groups, see "Apply Firewall Filters to Interfaces" on page 169.
packet-length *bytes*	Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.

Match Condition	Description
port *number*	TCP or UDP source or destination port field. You cannot specify both the port match and either the destination-port or source-port match conditions in the same term. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the text synonyms listed under destination-port.
precedence *ip-precedence-field*	IP precedence field. In place of the numeric field value, you can specify one of the following text synonyms (the field values are also listed): critical-ecp (0xa0), flash (0x60), flash-override (0x80), immediate (0x40), internet-control (0xc0), net-control (0xe0), priority (0x20), or routine (0x00).
protocol *number*	IP protocol field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): ah, egp (8), esp (50), gre (47), icmp (1), igmp (2), ipip (4), ipv6 (41), ospf (89), pim (103), rsvp (46), tcp (6), or udp (17).
source-port *number*	TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric field, you can specify one of the text synonyms listed under destination-port.

Table 24: Numeric Range IPv6 Firewall Filter Match Conditions

Match Condition	Description
address *address*	A 128-bit address that supports the standard syntax for IPv6 addresses. For more information, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.
destination-address *address*	A 128-bit address that is the final destination node address for the packet. The filter description syntax supports the text representations for addresses as described in RFC 2373 for IPv6 addresses. For more information about IPv6 address syntax, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.
destination-port *number*	TCP or UDP destination port field. You cannot specify both the port and destination-port match conditions in the same term. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the port numbers are also listed): afs (1483), bgp (179), biff (512), bootpc (68), bootps (67), cmd (514), cvspserver (2401), dhcp (67), domain (53), eklogin (2105), ekshell (2106), exec (512), finger (79), ftp (21), ftp-data (20), http (80), https (443), ident (113), imap (143), kerberos-sec (88), klogin (543), kpasswd (761), krb-prop (754), krbupdate (760), kshell (544), ldap (389), login (513), mobileip-agent (434), mobilip-mn (435), msdp (639), netbios-dgm (138), netbios-ns (137), netbios-ssn (139), nfsd (2049), nntp (119), ntalk (518), ntp (123), pop3 (110), pptp (1723), printer (515), radacct (1813), radius (1812), rip (520), rkinit (2108), smtp (25), snmp (161), snmptrap (162), snpp (444), socks (1080), ssh (22), sunrpc (111), syslog (514), tacacs-ds (65), talk (517), telnet (23), tftp (69), timed (525), who (513), xdmcp (177), zephyr-clt (2103), or zephyr-hm (2104).
icmp-code *number*	ICMP code field. This value or keyword provides more specific information than icmp-type. Because the value's meaning depends upon the associated icmp-type, you must specify icmp-type along with icmp-code. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated: parameter-problem: ip-header-bad (0), required-option-missing (1) redirect: redirect-for-host (1), redirect-for-network (0), redirect-for-tos-and-host (3), redirect-for-tos-and-net (2) time-exceeded: ttl-eq-zero-during-reassembly (1), ttl-eq-zero-during-transit (0) unreachable: communication-prohibited-by-filtering (13), destination-host-prohibited (10), destination-host-unknown (7), destination-network-prohibited (9), destination-network-unknown (6), fragmentation-needed (4), host-precedence-violation (14), host-unreachable (1), host-unreachable-for-TOS (12), network-unreachable (0), network-unreachable-for-TOS (11), port-unreachable (3), precedence-cutoff-in-effect (15), protocol-unreachable (2), source-host-isolated (8), source-route-failed (5)
icmp-type *number*	ICMP packet type field. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): echo-reply (0), echo-request (8), info-reply (16), info-request (15), mask-request (17), mask-reply (18), parameter-problem (12), redirect (5), router-advertisement (9), router-solicit (10), source-quench (4), time-exceeded (11), timestamp (13), timestamp-reply (14), or unreachable (3).

Match Condition	Description
interface-group *group-number*	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For information about configuration interface groups, see "Apply Firewall Filters to Interfaces" on page 169.
next-header *bytes*	An eight-bit IP protocol field that identifies the type of header immediately following the IPv6 header. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): egp (8), esp (50), gre (47), icmp (1), igmp (2), ipip (4), ipv6 (41), ospf (89), pim (103), rsvp (46), tcp (6), or udp (17).
packet-length *bytes*	Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.
port *number*	TCP or UDP source or destination port field. You cannot specify both the port match and either the destination-port or source-port match conditions in the same term. Typically, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the text synonyms listed under destination-port.
source-address *address*	Address of the source node sending the packet; 128 bits in length. The filter description syntax supports the text representations for addresses as described in RFC 2373 for IPv6. For more information about IPv6 address syntax, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.
source-port *number*	TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric field, you can specify one of the text synonyms listed under destination-port.
traffic-class *number*	An eight-bit field that specifies the class-of-service (CoS) priority of the packet. This field was previously used as the ToS field in IPv4. However, the semantics of this field (for example, DiffServ code points) are identical to IPv4.

Specify Address Filter Match Conditions

Address filter conditions match prefix values in a packet, such as IP source and destination prefixes. For address filter match conditions, you specify a keyword that identifies the field and one or more prefixes of that type that a packet must match. Table 25 describes the address filter match conditions.

You can specify the address in one of the following ways:

- Single prefix. A match occurs if the value of the field matches the prefix. For example:

  ```
  [edit firewall family family-name filter filter-name term term-name from]
  destination-address 10.0.0.0/8;
  ```

- Multiple prefixes. A match occurs if any one of the prefixes in the list matches the packet. For example:

  ```
  [edit firewall family family-name filter filter-name term term-name from]
  destination-address {
     10.0.0.0/8;
     192.168.0.0/32;
  }
  ```

  The order in which you list prefixes in the list is not significant. They are all evaluated to determine whether a match occurs. If prefixes overlap, longest-match rules are used to determine whether a match occurs. Each list of prefixes contains an implicit 0/0 except statement, which means that any prefix that does not match any prefix in the list is explicitly considered not to match.

To specify the address prefix, use the notation *prefix*/*prefix-length*. If you omit *prefix-length*, it defaults to /32. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@host# set destination-address 10
[edit firewall family family-name filter filter-name term term-name from]
user@host# show
destination-address {
    10.0.0.0/32;
}
```

To exclude a prefix, specify the string except after the prefix. In the following example, any addresses that fall under 131.0.0.0/8 match, except for addresses that fall under 131.108.0.0/16. All other addresses implicitly do not match this condition.

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
    131.108.0.0/16 except;
    131.0.0.0/8;
}
```

To match all destinations except one, in this example 10.1.1.0/24, configure the match conditions as follows:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
    0.0.0.0/0;
    10.1.1.0/24 except;
}
```

Because the prefixes are order-independent and use longest-match rules, longer prefixes subsume shorter ones as long as they are the same type (whether you specify except or not). This is because anything that would match the longer prefix would also match the shorter one. In the following example:

- 8.4.1.2 matches the 8.0.0.0/10 prefix, and thus the action in the then statement is taken.
- 8.2.1.2 matches the 8.2.0.0/16 prefix. Because this prefix is negated (that is, marked as except), an explicit mismatch occurs. The next term in the filter is evaluated, if there is one. If there are no more terms, the packet is discarded.
- 1.2.3.4 does not match any of the prefixes included in the source-address condition. Instead, it matches the implicit 0.0.0.0/0 except at the end of the list, and is considered to be a mismatch.

- The 8.3.0.0/16 statement is ignored because it falls under the address 8.0.0.0/10—both are the same type.
- The 10.2.2.2 except statement is ignored because it is subsumed by the implicit 0.0.0.0/0 except statement at the end of the list.

```
[edit firewall family family-name filter filter-name term term-name from]
source-address {
    8.0.0.0/10;
    8.2.0.0/16 except;
    192.168.1.0;
    192.168.1.192/26 except;
    192.168.1.254;
    8.3.0.0/16;                 # ignored
    10.2.2.2 except;            # ignored
}
```

Table 25: Address Firewall Filter Match Conditions

Match Condition	Description
address *prefix*	IP source or destination address field. You cannot specify both the address and the destination-address or source-address match conditions in the same term.
destination-address *prefix*	IP destination address field. You cannot specify the destination-address and address match conditions in the same term.
destination-prefix-list *prefix-list*	IP destination prefix list field. You cannot specify the destination-prefix-list and prefix-list match conditions in the same term.
prefix-list *prefix-list*	IP source or destination prefix list field. You cannot specify both the prefix-list and the destination-prefix-list or source-prefix-list match conditions in the same term.
source-address *prefix*	IP source address field. You cannot specify the source-address and address match conditions in the same rule.
source-prefix-list *prefix-list*	IP source prefix list field. You cannot specify the source-prefix-list and prefix-list match conditions in the same term.

You can also define a list of IP address prefixes under a *prefix-list* alias for frequent reference. You make this definition at the [edit policy-options] hierarchy level:

```
[edit policy-options]
    prefix-list prefix-list {
        address;
        address;
        address;
    }
}
```

Once you have defined a prefix list, you can use it when defining firewall filters:

```
[edit firewall family family-name filter filter-name term term-name]
from {
    source-prefix-list {
        prefix-list1;
        prefix-list2;
    }
    destination-prefix-list {
        prefix-list1;
    }
}
```

You can specify noncontiguous address prefixes in a filter term for firewall filters. Noncontiguous address prefixes are prefixes that are not adjacent or neighboring to one another. For example, in the following example, the following prefixes are noncontiguous: 0.0.0.10/0.0.0.255, 0.10.0.10/0.255.0.255, and 0.12.10.9/0.255.255.255:

```
[edit firewall family inet filter filter-name]
term term-name {
    address 0.0.0.10/0.0.0.255;
    destination-address 0.10.0.10/0.255.0.255;
    source-address 0.12.10.9/0.255.255.255 except;
}
```

Noncontiguous address prefixes are valid only for IPv4 filters. IPv6 filters do not support noncontiguous address prefixes.

You can also specify a netmask value rather than a prefix length, for example:

```
[edit firewall family inet filter filter-name]
term term-name {
    address 10.0.0.10/255.0.0.255;
}
```

The prefix notation shown matches any address with a first and last octet of 10. The address and netmask are separated by a forward slash (/). The second and third bytes of the prefix can be any value between 0 and 255.

For more information about prefixes, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

Specify Multiple Match Conditions

A complication arises with filters that specify both *address* and *port* matches:

- An address match occurs if either the source or destination address in the packet matches one of the prefixes in the list.
- A port match occurs if either the source or destination port in the packet matches one of the port ranges in the list.

For example, you could apply the following terms within a firewall filter:

```
[edit firewall family family-name filter filter-name]
term 1 {
    from {
        address {
            1.0.0.0/12;
        }
        protocol tcp;
    }
    then {
        count lf1-1;
        accept;
    }
}
```

```
term 2 {
    from {
        address {
            1.26.0.0/15;
        }
        protocol tcp;
    }
    then {
        count lf1-2;
        accept;
    }
}
```

A packet whose source address is 1.0.1.1 and whose destination address is 1.27.1.1 should increment the first counter, as should a packet with source address 1.27.1.1 and destination address 1.0.1.1. Sometimes, however, one of these packets increments the second counter rather than the first.

The problem is that the address matches are seen as mutually exclusive alternatives, which are compiled into a form that evaluates the match in parallel without regard to term ordering. This works for single-field matches such as source-address or destination-address, but not for multiple-field matches where there is no mutual exclusivity. It still produces correct matches in this case (the packet always matches the from condition in the term whose action it takes), but loses the term ordering that should be used to distinguish multiple matches.


Note

As a workaround in this situation, avoid using address and port for match conditions. You can use source-address, destination-address, source-port, or destination-port instead.

If the application absolutely requires matching the same prefix against either source-address or destination-address, write two terms in sequence, for example:

```
term 1 {
    from {
        source-address 192.168/16;
    }
    then accept;
}
term 2 {
    from {
        destination-address 192.168/16;
    }
    then accept;
}
```

Specify Bit-Field Filter Match Conditions (IPv4 Traffic Only)

Bit-field filter conditions match packet fields if particular bits in those fields are or are not set. You can match the IP options, TCP flags, and IP fragmentation fields. For bit-field filter match conditions, you specify a keyword that identifies the field and tests to determine that the option is present in the field. Table 26 describes the bit-field match conditions.

To specify the bit-field value to match, enclose the value in quotation marks (double quotes). For example, a match occurs if the RST bit in the TCP flags field is set:

```
tcp-flags “rst”;
```

Generally, you specify the bits being tested using keywords. Bit-field match keywords always map to a single bit value. You also can specify bit fields as hexadecimal or decimal numbers.

To negate a match, precede the value with an exclamation point. For example, a match occurs only if the RST bit in the TCP flags field is *not* set:

```
tcp-flags “!rst”;
```

To match multiple bit-field values, use the logical operators list in Table 27. The operators are listed in order, from highest precedence to lowest precedence. Operations are left-associative.

As an example of a logical AND operation, in the following, a match occurs if the packet is the initial packet on a TCP session:

```
tcp-flags “syn & !ack”;
```

As an example of a logical OR operation, in the following, a match occurs if the packet is *not* the initial packet on a TCP session:

```
tcp-flags “!syn | ack”;
```

As an example of grouping, in the following, a match occurs for any packet that is either a TCP reset or is not the initial packet in the session:

```
tcp-flags “!(syn & !ack) | rst”;
```

When you specify a numeric value that has more than one bit set, the value is treated as a logical AND of the set bits. For example, the following two values are the same and a match occurs only if either bit 0x01 or 0x02 is not set:

```
tcp-flags “!0x3”;
tcp-flags “!(0x01 & 0x02)”;
```

You can use text synonyms to specify some common bit-field matches. You specify these matches as a single keyword. For example:

```
tcp-established;
```

Table 26: Bit-Field Firewall Filter Match Conditions

Match Condition	Description
Conditions with Variables	
fragment-flags *number*	IP fragmentation flags. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): dont-fragment (0x4000), more-fragments (0x2000), or reserved (0x8000).
ip-options *number*	IP options. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): loose-source-route (131), record-route (7), router-alert (148), strict-source-route (137), or timestamp (68).
tcp-flags *number*	TCP flags. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more details, see "How Firewall Filters Test a Packet's Protocol" on page 161. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): ack (0x10), fin (0x01), push (0x08), rst (0x04), syn (0x02), or urgent (0x20).
Text Synonyms	
first-fragment	First fragment of a fragmented packet. This condition does not match unfragmented packets.
is-fragment	This condition matches if the packet is a trailing fragment; it does not match the first fragment of a fragmented packet. To match both first and trailing fragments, you can use two terms, or you can use "fragment-range 0-8191".
tcp-established	TCP packets other than the first packet of a connection. This is a synonym for "(ack | rst)". This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.
tcp-initial	First TCP packet of a connection. This is a synonym for "(syn & !ack)". This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.

Table 27: Bit-Field Logical Operators

Logical Operator	Description
(…)	Grouping
!	Negation
& or +	Logical AND
| or ,	Logical OR

Specify Class-based Filter Match Conditions

Class-based filter conditions match packet fields based on source class or destination class. A source class is a set of source prefixes grouped together and given a class name. A destination class is a set of destination prefixes grouped together and given a class name.

You can specify the source class in the following way:

source-class *class-name*;

You can specify the destination class in the following way:

destination-class *class-name*;


Note

Class-based filter match conditions are supported for inet and inet6 address families on the M-series platforms.

How Firewall Filters Test a Packet's Protocol

If you specify a port match condition or a match of the ICMP type or TCP flags field, there is no implied protocol match. If you use one of the following match conditions in a term, you should explicitly specify the protocol in the same term:

- destination-port—Specify the match protocol tcp or protocol udp in the same term.
- icmp-code—Specify the match protocol icmp in the same term.
- icmp-type—Specify the match protocol icmp in the same term.
- port—Specify the match protocol tcp or protocol udp in the same term.
- source-port—Specify the match protocol tcp or protocol udp in the same term.
- tcp-flags—Specify the match protocol tcp in the same term.

When examining match conditions, the policy framework software tests only the specified field itself. The software does not also test the IP header to determine that the packet is indeed an IP packet.

If you do not explicitly specify the protocol, when using the fields listed above, design your filters carefully to ensure that they are performing the expected matches. If you specify a match of destination-port ssh, the policy framework software deterministically matches any packets that have a value of 22 in the 2-byte field that is 2 bytes beyond the end of the IP header, without ever checking the IP protocol field.

Example: Do Not Test Packet Protocol

The first term matches all packets except for TCP and UDP packets, so only TCP and UDP packets are evaluated by third term (term test-a-port):

```
[edit]
firewall {
    family inet {
        filter test-filter {
            term all-but-tcp-and-udp {
                from {
                    protocol-except [tcp udp];
                }
                then accept;
            }
            term test-an-address {
                from {
                    address 192.168/16;
                }
                then reject;
            }
            term test-a-port {
                from {
                    destination-port [ssh dns];
                }
                then accept;
            }
            term dump-anything-else {
                then reject;
            }
        }
    }
}
```

Examples: Define Firewall Filters

Block telnet and secure shell (ssh) access to all but the 192.168.1.0/24 subnet. This filter also logs any ssh or telnet traffic attempts from other subnets to the firewall log buffer:

```
[edit]
firewall {
    family inet {
        filter local-access-control {
            term terminal-access {
                from {
                    address {
                        192.168.1.0/24;
                    }
                    protocol tcp;
                    port [ssh telnet];
                }
                then accept;
            }
            term terminal-access-denied {
                from {
                    protocol tcp;
                    port [ssh telnet];
                }
                then {
                    log;
                    reject;
                }
            }
            term default-term {
                then accept;
            }
        }
    }
}
```

Block Trivial File Transfer Protocol (TFTP) access, logging any attempts to establish TFTP connections:

```
[edit]
firewall {
    family inet {
        filter tftp-access-control {
            from {
                protocol udp;
                port tftp;
            }
            then {
                log;
                discard;
            }
        }
    }
}
```

By default, to decrease vulnerability to denial-of-service (DoS) attacks, the JUNOS software filters and discards Dynamic Host Configuration Protocol (DHCP) or Bootstrap Protocol (BOOTP) packets that have a source address of 0.0.0.0 and a destination address of 255.255.255.255. This default filter is known as a unicast reverse path forwarding (unicast RPF) check. However, some vendors' equipment automatically accepts these packets. To interoperate with other vendors' equipment, you can configure a filter that checks for both these addresses and overrides the default RPF-check filter by accepting these packets.

Configure a filter (rpf-dhcp) that accepts DHCP packets with a source address of 0.0.0.0 and a destination address of 255.255.255.255:

```
[edit firewall family inet]
filter rpf-dhcp {
    term dhcp {
        from {
            source-address {
                0.0.0.0/32;
            }
            destination-address {
                255.255.255.255/32;
            }
        }
        then {
            accept;
        }
    }
}
```

To apply this filter to an interface, include the rpf-check fail-filter statement at the [edit interface *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```
[edit interface interface-name unit logical-unit-number family inet]
rpf-check fail-filter rpf-dhcp;
```

Define a policer for a destination class class1:

```
[edit]
firewall {
    family inet {
        filter filter1 {
            policer police-class1
                if-exceeding {
                    bandwidth-limit 25;
                    burst-size-limit 1000;
                }
                then {
                    discard;
                }
            }
            term term1 {
                from {
                    destination-class class1;
                }
                then {
                    policer police-class1;
                }
            }
        }
    }
}
```

Count individual IP option packets, but do not block any traffic. Also, log packets that have loose or strict source routing:

```
[edit]
firewall {
    family inet {
        filter ip-option-filter {
            term match-strictsource {
                from {
                    ip-options strict-source-route;
                }
                then {
                    count strict-source-route;
                    log;
                    accept;
                }
            }
            term match-loose-source {
                from {
                    ip-options loose-source-route;
                }
                then {
                    count loose-source-route;
                    log;
                    accept;
                }
            }
            term match-record {
                from {
                    ip-options record-route;
                }
                then {
                    count record-route;
                    accept;
                }
            }
            term match-timestamp {
                from {
                    ip-options timestamp;
                }
                then {
                    count timestamp;
                    accept;
                }
            }
            term match-router-alert {
                from {
                    ip-options router-alert;
                }
                then {
                    count router-alert;
                    accept;
                }
            }
            term match-all {
                then accept;
            }
        }
    }
}
```

Accept only Open Shortest Path First (OSPF) packets from an address in the prefix 131.108.0.0/16, discarding all other packets with an administratively-prohibited ICMP message:

```
[edit]
firewall {
    family inet {
        filter ospf-filter {
            term term1 {
                from {
                    source-address {
                        131.108.0.0/16;
                    }
                    protocol ospf;
                }
            }
            term default-term {
                then {
                    reject administratively-prohibited;        # default reject action
                }
            }
        }
    }
}
```

Match packets that are either OSPF packets or packets that come from an address in the prefix 131.108/16, and send an administratively-prohibited ICMP message for all packets that do not match:

```
[edit]
firewall {
    family inet {
        filter ospf-or-131 {
            term protocol-match {
                from {
                    protocol ospf;
                }
            }
            term address-match {
                from {
                    source-address {
                        131.108.0.0/16;
                    }
                }
            }
        }
    }
}
```

Reject all addresses except 192.168.5.0/24. In the first term, the statement 192.168.5.2/24 except causes this address to be considered a mismatch and this address is passed to the next term in the filter. The address 0.0.0.0/0 in the first term matches all other packets, and these are counted, logged, and rejected. In the second term, all packets that passed though the first term (that is, packets whose address matches 192.168.5.2/24) are counted, logged, and accepted.

```
[edit]
firewall {
    family inet {
        filter fire1 {
            term 1 {
                from {
                    address {
                        192.168.5.0/24 except;
                        0.0.0.0/0;
                    }
                }
                then {
                    count reject-pref1-1;
                    log;
                    reject;
                }
            }
            term 2 {
                then {
                    count reject-pref1-2;
                    log;
                    accept;
                }
            }
        }
    }
}
```

Block all TCP connection attempts to port 179 from all places except the configured Border Gateway Protocol (BGP) peers:

```
[edit]
firewall {
    family inet {
        filter bgp179 {
            term 1 {
                from {
                    source-address {
                        0.0.0.0/0;
                    }
                    source-prefix-list {
                        bgp179 except;
                    }
                    destination-port bgp;
                }
                then {
                    reject;
                }
            }

            term 2 {
                then {
                    accept;
                }
            }
        }
    }
}
```

Expand the prefix list bgp179 to include all BGP group neighbors:

```
[edit policy-options]
prefix-list bgp179 {
    apply-path “protocols bgp group <*> neighbor <*>”;
}
```

Apply the filter bgp179 to interface lo0:

```
[edit interfaces lo0]
root@canopy# show
unit 0 {
    family inet {
        filter {
            input bgp179;
        }
        address 127.0.0.1/32;
    }
}
```

Apply Firewall Filters to Interfaces

For a firewall filter to work, you must apply it to at least one interface. To do this, include the filter statement when configuring the logical interface at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name]
filter {
    input filter-name;
    output filter-name;
}
```

In the input statement, list the name of one firewall filter to be evaluated when packets are received on the interface. Input filters applied to the loopback interface, lo0, affect only inbound traffic destined for the Routing Engine.

In the output statement, list the name of one firewall filter to be evaluated when packets are transmitted on the interface. Output filters applied to the loopback interface, lo0, affect only outbound traffic sent from the Routing Engine.

You can apply only one input and one output firewall filter to each interface. You can use the same filter one or more times.

When you apply a filter to an interface, it is evaluated against all the data packets passing through that interface. The exception is the loopback interface, lo0, which is the interface to the Routing Engine and carries no data packets. If you apply a filter to the lo0 interface, the filter affects the local packets received or transmitted by the Routing Engine.

Filters apply to all packets entering an interface, not just the packets destined for the Routing Engine. To filter packets destined for the Routing Engine, configure the group statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name* filter] hierarchy level. For more information, see “Define Interface Groups” on page 170.

For filters applied to data packets to function, the router must contain an Internet Processor II ASIC.

You can configure the following additional properties when applying filters to interfaces:

- Configure Interface-Specific Counters on page 169
- Define Interface Groups on page 170

Configure Interface-Specific Counters

When you configure a firewall filter that is applied to multiple interfaces, you can name individual counters specific to each interface. These counters enable you to easily maintain statistics on the traffic transiting the different interfaces.

Configuration of interface-specific counters also creates separate instances of any policers you have configured for the same interface. For more information about policers, see “Policer Configuration” on page 185.

To configure interface-specific counters, include the interface-specific statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level:

```
[edit firewall family family-name filter filter-name]
interface-specific;
```

The counter name is restricted to 24 bytes. If the renamed counter exceeds this maximum length, the policy framework software might reject it.

Example: Configure Interface-Specific Counters

Configure an interface-specific counter:

```
[edit firewall]
    family inet {
        filter test {
            interface-specific;
            term 1 {
                from {
                    address {
                        1.0.0.0/12;
                    }
                    protocol tcp;
                }
                then {
                    count sample1;
                    accept;
                }
            }
        }
    }
}
```

When you apply this filter to the input interface of at-1/1/1.0 and the output interface of so-2/2/2.2, the counters are named sample1-at-1/1/1.0-i and sample1-so-2/2/2/.2-o. The suffixes -i (input) and -o (output) are added to the counter names automatically.

Define Interface Groups

When applying a firewall filter, you can define an interface to be part of an *interface group*. Packets received on that interface are tagged as being part of the group. You then can match these packets using the interface-group match statement, as described in Table 23 on page 152.

To define an interface to be part of an interface group, include the group statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name* filter] hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name filter]
group group-number;
input filter-name;
output filter-name;
```

In the group statement, specify the interface group number to be associated with the filter.

In the input statement, list the name of one firewall filter to be evaluated when packets are received on the interface.

In the output statement, list the name of one firewall filter to be evaluated when packets are transmitted on the interface.

Example: Define Interface Groups

Create a filter that contains an interface group:

```
[edit firewall]
family inet {
    filter if-group {
        term group1 {
            from {
                interface-group 1;
                address {
                    207.79.80.114/32;
                }
                protocol tcp;
                port finger;
            }
            then {
                count if-group-counter1;
                log;
                reject;
            }
        }
        term group-2 {
            then {
                count if-group-counter2;
                log;
                accept;
            }
        }
    }
}
```

Assign one or more interfaces to the interface group referenced in the filter:

```
[edit interfaces]
fxp0 {
    unit 0 {
        family inet {
            filter {
                group 1;
            }
            address 192.168.5.38/24;
        }
    }
}
```

Apply the filter that contains an interface group:

```
[edit interfaces]
family inet {
    lo0 {
        unit 0 {
            family inet {
                filter {
                    input if-group;
                    group 1;
                }
                address 127.0.0.1/32;
                address 192.168.77.1/32;
            }
        }
    }
}
```

Configure Accounting

Juniper Networks routers can collect various kinds of data about traffic passing through the router. You can set up one or more *accounting profiles* that specify some common characteristics of this data, including the following:

- Fields used in the accounting records
- Number of files that the router retains before discarding, and the number of bytes per file
- Polling period that the system uses to record the data

To implement an accounting profile, you must configure the profile and then apply it to an interface or firewall filter. For more information, see the *JUNOS Internet Software Configuration Guide: Network Management*.

Configure a Firewall Filter Accounting Profile

There are several types of accounting profiles: interface, firewall filter, destination class, and Routing Engine. To configure an accounting profile, include statements at the [edit accounting-options] hierarchy level. To activate firewall filter profiles, you must reference them at the [edit firewall family *family-name*] hierarchy level. If you reference the same profile name from both a firewall filter and an interface in the same configuration, it causes an error.

The following example shows accounting profile fw_profile for the firewall filter myfilter. For more information about configuring accounting profiles, see the *JUNOS Internet Software Configuration Guide: Network Management*:

```
[edit]
accounting-options {
    filter-profile fw_profile {
        file fw_accounting;
        interval 60;
        counters {
            counter1;
            counter2;
            counter3;
        }
    }
}
```

To apply the fw_profile accounting profile to a firewall filter, include the accounting-profile statement as shown:

```
[edit]
firewall {
    family inet {
        filter myfilter {
            accounting-profile fw_profile;
            ...
            term accept-all {
                then {
                    count counter1;
                    accept;
                }
            }
        }
    }
}
```

Configure Filter-Based Forwarding

You can configure filters to classify packets based on source address and specify the forwarding path the packets take within the router. For example, you can use this filter for applications to differentiate traffic from two clients that have a common access layer (for example, a Layer 2 switch) but are connected to different Internet service providers (ISPs). When the filter is applied, the router can differentiate the two traffic streams and direct each to the appropriate network. Depending on the media type the client is using, the filter can use the source IP address to forward the traffic to the corresponding network through a tunnel. You can also configure filters to classify packets based on IP protocol type or IP precedence bits.


Note

You can forward packets based on input filters only; you cannot forward packets based on output filters.

Filter-based forwarding is supported for Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6).

To direct traffic meeting defined match conditions to a specific routing instance, include the routing-instance filter action:

routing-instance *routing-instance*;

For IPv4 traffic, include the action at the [edit firewall family inet filter *filter-name* term *term-name* then] hierarchy level. For IPv6 traffic, include the action at the [edit firewall family inet6 filter *filter-name* term *term-name* then] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name* then] hierarchy level.

The routing-instance filter action accepts the traffic meeting the match conditions and directs it to the routing instance named in *routing-instance*. For information about forwarding instances and routing instances, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

To complete the configuration, you must also create a routing table group that adds interface routes to the following routing instances:

- Routing instance named in the action
- Default routing table inet.0

You create a routing table group to resolve the routes installed in the routing instance to directly connected next hops on that interface. For more information on routing table groups and interface routes, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

Examples: Configure Filter-Based Forwarding

Configure a filter to direct traffic to ISP1 or ISP2 based on source address matching:

```
[edit firewall]
family inet {
    filter classify-customers {
        term isp1-customers {
            from {
                source-address 10.1.1.0/24;
                source-address 10.1.2.0/24;
            }
            then {
                routing-instance isp1-route-table;
            }
        }
        term isp2-customers {
            from {
                source-address 10.2.1.0/24;
                source-address 10.2.2.0/24;
            }
            then {
                routing-instance isp2-route-table;
            }
        }
        term default {
            then {
                accept;
            }
        }
    }
}
```

Configure a FBF filter for family inet6:

```
[edit]
firewall {
    family inet {
        filter ftf_fbf {
            term 0 {
                from {
                    source-address {
                        ::10.34.1.0/120;
                    }
                }
                then {
                    count ce1;
                    log;
                    routing-instance ce1;
                }
            }
            term 1 {
                from {
                    source-address {
                        ::10.34.2.0/120;
                    }
                }
                then {
                    count ce2;
                    log;
                    routing-instance ce2;
                }
            }
            term default {
                then {
                    count default;
                    accept;
                }
            }
        }
    }
}
```

Configure Forwarding Table Filters

The following section describes how to configure forwarding table filters.

Overview

You set up a forwarding table filter in essentially the same way a firewall filter: you define it, then you apply it. However, you apply the filters differently:

- Instead of applying a forwarding table filter to an interface, you apply it to a forwarding table, which is associated with a routing instance and a Virtual Private Network (VPN).
- Instead of applying input and output firewall filters, by default, you can apply an input forwarding table filter only.

All packets are subjected to the input forwarding table filter that applies to the forwarding table. A forwarding table filter controls which packets the router accepts and performs a forwarding table lookup for, thereby controlling which packets the router forwards on the interfaces.

When the router receives a packet, it determines where to forward the packet by looking in a forwarding table, which is associated with the VPN on which the packet will be sent, for the best route to the destination. The router then forwards the packet toward its destination through the appropriate interface.

Configure a Forwarding Table Filter

A forwarding table filter allows you to filter data packets based on their components and to perform an action on packets that match the filter.

To configure a forwarding table filter, do the following:

1. Define a forwarding table filter:

    a. Configure the family address type: IPv4 (inet), IPv6 (inet6), or MPLS (mpls).

    b. Define one or more *terms*, which are named structures in which match conditions and actions are defined.

    c. Define a *match condition*, which is the criterion against which a bearer packet is compared; for example, the IP address of a source device or a destination device. You can specify multiple criteria in a match condition.

    d. Define an *action*, which is what happens if all criteria match; for example, the GGSN accepting the bearer packet, performing a lookup in the forwarding table, and forwarding the packet to its destination; discarding the packet; and discarding the packet and returning a rejection message. In addition to an action, you can define one or more *action modifiers*, which are actions that are taken in addition to the GGSN accepting or discarding a packet when all criteria match; for example, counting the packets and logging a packet.

For more information about configuring firewall filters, see “Configure Firewall Filters” on page 143.

2. Apply the forwarding table filter as an input filter to a forwarding table. The forwarding table filter controls which bearer packets the router accepts and forwards.

To define a forwarding table filter, include the firewall statement at the [edit] hierarchy level:

```
[edit]
firewall {
    family family-name {
        filter filter-name {
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    action;
                    action-modifiers;
                }
            }
        }
    }
}
```

To create a forwarding table, include the instance-type statement at the [edit routing-instance] hierarchy level:

```
[edit]
routing-instance instance-name {
    instance-type forwarding;
}
```

To apply a forwarding table filter to a VRF forwarding table, include the filter input statement at the [edit routing-instance *instance-name* forwarding-option family *family*] hierarchy level:

```
[edit]
routing-instance routing-instance-name {
    instance-type forwarding;
    forwarding-options {
        family family {
            filter {
                input filter-name;
            }
        }
    }
}
```

To apply a forwarding table filter to a forwarding table, include the filter input statement at the [edit forwarding-option family *family*] hierarchy level:

```
[edit]
forwarding-option {
    family family {
        filter {
            input filter-name;
        }
    }
}
```

To apply a forwarding table filter to the default forwarding table inet.0, which is not associated with a specific routing instance, include the filter input statement at the [edit routing-options] hierarchy level:

```
[edit]
routing-options {
    rib inet.0 {
        filter {
            input filter-name;
        }
    }
}
```

For information about the routing-instance and routing-options statements, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols.*

Configure Firewall Filter System Logging

System logging can be configured for the firewall filter process. You can set system logging to record messages of a particular level or all levels. The messages are sent to a system logging file.

The following is a sample system logging configuration for the firewall filter icmp-syslog. For more information about configuring system logging, see the *JUNOS Internet Software Configuration Guide: Getting Started*.

```
[edit]
system {
    syslog {
        file filter {
            firewall any;
            archive no-world-readable;
        }
    }
}
```

This causes the syslog daemon to write any messages with the syslog facility of firewall to the file /var/log/filter. This keeps the messages out of the main system log file and makes them easier to find.

Example: Configure Firewall Filter System Logging

Create a filter that logs and counts ICMP packets that have 192.168.207.222 as either their source or destination:

```
[edit]
firewall {
    family inet {
      filter icmp-syslog {
        term icmp-match {
            from {
            address {
                192.168.207.222/32;
            }
            protocol icmp;
          }
           then {
            count packets;
            syslog;
            accept;
          }
        }
        term default {
            then accept;
        }
      }
    }
}
```

Enter the show log filter command to display the results:

```
root@systech> show log filter
Mar 20 08:03:11 systech feb FW: so-0/1/0.0   A icmp 192.168.207.222 192.168.207.223    0
0 (1 packets)
```

This output file contains the following fields:

- Date and Time—Date and time at which the packet was received (not shown in the default).
- Filter action:
  - A—Accept (or next term)
  - D—Discard
  - R—Reject
- Protocol—Packet's protocol name or number.
- Source address—Source IP address in the packet.
- Destination address—Destination IP address in the packet.

If the protocol is ICMP, the ICMP type and code are displayed. For all other protocols, the source and destination ports are displayed.

The last two fields (both zero) are the source and destination TCP/UDP ports, respectively, and are shown for TCP or UDP packets only. This log message indicates that only one packet for this match has been detected in about a one-second interval. If packets arrive faster, the system log function compresses the information so that less output is generated, and displays an output similar to the following:

```
root@systech> show log filter
Mar 20 08:08:45 systech feb FW: so-0/1/0.0  A icmp 192.168.207.222 192.168.207.223    0
0 (515 packets)
```

Chapter 10
Policer Overview

Policing, or rate limiting, enables you to limit the amount of traffic that passes into or out of an interface. It is an essential component of firewall filters that is designed to thwart denial-of-service (DoS) attacks. Policing applies two types of rate limits on the traffic:

- Bandwidth—The number of bits per second permitted, on average.
- Maximum burst size—The maximum size permitted for bursts of data that exceed the given bandwidth limit.

Policing uses the *token-bucket algorithm*, which enforces a limit on average bandwidth while allowing bursts up to a specified maximum value. It offers more flexibility than the *leaky bucket algorithm* (see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*) in allowing a certain amount of bursty traffic before it starts discarding packets.

You can define specific classes of traffic on an interface and apply a set of rate limits to each. You can use a policer in one of two ways: as part of a filter configuration or as an individual policer statement that applies to each family on an interface.

After you have defined and named a policer, it is stored as a template. You can later use the same policer name to provide the same policer configuration each time you wish to use it. This eliminates the need to define the same policer values more than once.

Chapter 11
Policer Configuration

To configure policers, you include statements at the [edit firewall] hierarchy level of the configuration:

```
[edit]
firewall {
    policer policer-name {
        filter-specific;
        if-exceeding {
            bandwidth-limit bps;
            bandwidth-percent number;
            burst-size-limit bytes;
        }
        then {
            policer-action;
        }
    }
    family family-name {
        filter filter-name {
            accounting-profile name;
            interface-specific;
        }
        prefix-action name {
            count;
            destination-prefix-length prefix-length;
            policer policer-name;
            source-prefix-length prefix-length;
            subnet-prefix-length prefix-length;
        }
        prefix-policer {
            policer policer-name;
        }
    }
}
```

The following sections explain the tasks required for configuring policers and provide configuration examples:

Minimum Policer Configuration

To configure a policer, you must perform at least the following tasks:

- Configure policers—To configure policers, include the policer statement at the [edit firewall] hierarchy level. After policers are defined, you reference them in the then clause of a term:

```
[edit]
firewall {
    policer policer-name {
        filter-specific;
        if-exceeding {
            bandwidth-limit bps;
            bandwidth-percent number;
            burst-size-limit bytes;
        }
        then {
            policer-action;
        }
    }
    family family-name {
        filter filter-name  {
    }
}
```

- Add actions, such as accept, discard, or next term, or action modifiers, such as count or log.

- Apply the policers to an interface to activate them.

The policer is applied to the packet first, and if the packet exceeds the defined limits, the actions of the then clause of the policer are applied. If the result of the policing action is not a discard, the remaining components of the then clause of the term are applied.

To display statistics about a filter statement policer configuration, use the show policers command.

Configure Policers

To configure term-specific policers, include the policer statement:

```
policer policer-name {
    if-exceeding {
        bandwidth-limit rate;
        bandwidth-percent number;
        burst-size-limit bytes;
    }
    then {
        policer-action;
    }
}
```

You can configure the policer statement at the [edit firewall] hierarchy level. The following sections describe the components of the policer statement and provide policer configuration examples:

- Configure Rate Limiting on page 187
- Configure a Policer Action on page 188
- Configure Multifield Classification and Policing on page 189
- Configure Filter-Specific Policers on page 189
- Configure Prefix-Specific Action on page 190
- Examples: Classify Traffic on page 194

Configure Rate Limiting

To specify the rate limiting part of a policer, include an if-exceeding statement at the [edit firewall policer *policer-name*] hierarchy level:

```
[edit firewall policer policer-name]
if-exceeding {
    bandwidth-limit bps;
    bandwidth-percent number;
    burst-size-limit bytes;
}
```

You specify the bandwidth limit in bits per second. You can specify the value as a complete decimal number or as a decimal number followed by the abbreviation k (1000), m (1,000,000), or g (1,000,000,000). There is no absolute minimum value for bandwidth limit, but any value below 61,040 bps will result in an effective rate of 30,520 bps. The maximum bandwidth limit is 4.29 Gbps.

You can rate-limit based upon port speed. This port speed can be specified by a bandwidth percentage in a policer. You must specify the percentage as a complete decimal number between 1 and 100.

You cannot rate-limit based on bandwidth percentage for aggregate, tunnel, and software interfaces. The bandwidth percentage policer cannot be used for forwarding table filters. This can only be used for interface specific filters.

The maximum burst size controls the amount of traffic bursting allowed. To determine the value for the burst-size limit, the preferred method is to multiply the bandwidth of the interface on which you are applying the filter by the amount of time you allow a burst of traffic at that bandwidth to occur; for example, 5 milliseconds.

burst size = bandwidth x allowable time for burst traffic

If you do not know the interface bandwidth, you can multiply the maximum transmission unit (MTU) of the traffic on the interface by 10 to obtain a value. For example, the burst size for an MTU of 4700 would be 47,000 bytes. At minimum, burst size should be at least 10 interface MTUs. The maximum value for the burst-size limit is 100 MB.

For a sample filter configuration for rate limiting, see “Examples: Configure Policing” on page 196.

Configure a Policer Action

If a packet does not exceed its rate limits, it is processed further without being affected. If the packet exceeds its limits, it is handled in one of two ways, depending on what you specify:

- Discarded
- Marked for subsequent processing based on its loss priority and forwarding class

To configure a policer action, include the then statement:

```
then {
    policer-action;
}
```

You can configure policer actions at the [edit firewall policer *policer-name*] hierarchy level. Policer actions include the following:

- discard—Discard a packet that exceeds the rate limits.
- loss-priority—Set the loss priority level to low or high.
- forwarding-class—Specify the forwarding class to any class name already configured for the forwarding class.

Example: Configure a Policer Action

Discard any packet that exceeds a bandwidth of 300 Kbps and a burst-size limit of 500 KB:

```
[edit firewall]
policer p1 {
    if-exceeding {
        bandwidth-limit 300k;
        burst-size-limit 500k;
    }
    then {
        discard;
    }
}
```

Configure Multifield Classification and Policing

You can configure *multifield classifiers* within a firewall filter to set the packet's forwarding class and packet loss priority. You can also apply policers to packets matching some classification term. The policing action might affect the resulting forwarding class, packet loss priority, and accept or drop status. For more information, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

To configure the forwarding class and loss priority, include the forwarding-class *class-name* and loss-priority *level* actions at the [edit firewall filter *filter-name* term *term-name* then] or [edit firewall filter *filter-name* policer *policer-name* then] hierarchy level:

```
then {
    loss-priority level;
    forwarding-class class-name;
}
```

You can specify one or both of the following actions:

- loss-priority—Set the loss priority level to low or high.
- forwarding-class—Specify the forwarding class to any class name already configured for the forwarding class.

For more information about forwarding class and loss priority, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

Configure Filter-Specific Policers

You can configure filter-specific policers within the firewall configuration. Filter-specific policers allows you to configure policers and counters for a specific filter name.

To configure filter-specific policers, include the filter-specific statement at the [edit firewall policer *policer-name*] hierarchy level:

```
filter-specific;
```

If the filter-specific statement is not configured, then the policer defaults to a term-specific policer.

You can apply the filter-specific policers to the family inet.

To apply filter-specific policers, include the prefix-policer statement at the [edit firewall family inet] hierarchy level:

```
prefix-policer {
    policer policer-name;
}
```

When you configure this statement, the policer referenced is the filter-specific policer.

Configure Prefix-Specific Action

You can configure prefix-specific action within the firewall configuration. Prefix-specific action allows you to configure policers and counters for specific addresses or ranges of addresses. This allows you to essentially create policers and counters on a per-prefix level.

To configure prefix-specific actions, include the count, destination-prefix-length *prefix-length*, policer *policer-name*, source-prefix-length *prefix-length*, and subnet-prefix-length *prefix-length* statements at the [edit firewall family inet prefix-action *name*] hierarchy level:

```
prefix-action name {
    count;
    destination-prefix-length prefix-length;
    policer policer-name;
    source-prefix-length prefix-length;
    subnet-prefix-length prefix-length;
}
```

Prefix-specific action is supported for IPv4 inet address family.

The following section describes the statement options:

- count—Specify this option to enable a prefix-specific counter.
- destination-prefix-length—Refer to the destination address range specified for a prefix-specific policer or counter.
- policer—Specify this option to enable a set of prefix-specific policers.
- source-prefix-length—Refer to the source address range specified for a prefix-specific policer or counter.
- subnet-prefix-length—Refers to the total address range of the subnet supported.

The source or destination prefix length must be larger than the subnet prefix length.

Examples: Configure Prefix-Specific Actions

The following example creates a prefix-specific policer operating on the source address and applies it to the input interface.

```
[edit]
firewall {
    policer host-policer {
        filter-specific;
        if-exceeding {
            bandwidth-limit bps;
            burst-size-limit bytes;
        }
        then {
            discard;
        }
    }
    family inet {
        prefix-action ftp-policer-set {
            count;
            destination-prefix-length 32;
            policer host-policer;
            subnet-prefix-length 24;
        }
        filter filter-ftp {
            term term1{
                from {
                    destination-address 10.10.10/24;
                    destination-port ftp;
                }
                then {
                    prefix-action ftp-policer-set;
                }
            }
        }
    }
}
```

The following example filters all packets going to the /24 subnet, letting it pass to the prefix-specific action policers. In the policer set, the last octet of the source address field of the packet is used to index into the respective prefix-specific action policers.

```
[edit]
firewall {
policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
    }
    family inet {
        prefix-action per-source-policer {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
    }
```

```
    filter limit-all-hosts {
        term one {
            from {
                source-address {
                    10.10.10.0/24;
                }
            }
            then prefix-action per-source-policer;
        }
    }
}
```

The following example filters all packets, letting it pass to the prefix-specific action policers. In the policer set, the last octet of source address field of the packet is used to index into the corresponding prefix-specific action policers.

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
    }
    family inet {
        prefix-action per-source-policer {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
    }
    filter limit-all-hosts {
        term one {
            then prefix-action per-source-policer;
        }
    }
}
```

In the following example, packets belonging to the 10.10.10.0/24 subnet are subjected to policing by the prefix-specific action policers.There are 128 policers defined in the policer set. Therefore the /24 subnet can be thought of being split into two /25s subnets, both of them sharing the same prefix-specific action set.

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
    }
    family inet {
        prefix-action per-source-policer {
            policer 1Mbps-policer;
            subnet-prefix-length 25;
            source-prefix-length 32;
        }
    }
```

```
    filter limit-all-hosts {
        term one {
            from {
                source-address {
                    10.10.10.0/24;
                }
            }
            then prefix-action per-source-policer;
        }
    }
}
```

The following example defines 256 policers based on the last octet of the source address field. However, you are only allowing a subset of that to pass through the match condition. As a result, only the lower half of the set is used.

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
    }
    family inet {
        prefix-action per-source-policer {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
    }
    filter limit-all-hosts {
        term one {
            from {
                source-address {
                    10.10.10.0/25;
                }
            }
            then prefix-action per-source-policer;
        }
    }
}
```

The following example accepts packets from 10.10.10/24 and 11.11/16 subnets and subject them to policing by the same set of prefix-specific action policers. The policers are shared by packets across both subnets. There is a one to one correspondence between the 10.10.10/24 subnet. For the 11.11/16 subnet, there is a many to one correspondence. Here, each of the 11.11.0/24, 11.11.1/24, 11.11.2/24 ...11.11.255/24 share the same prefix-specific action set.

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
    }
```

```
    family inet {
        prefix-action per-source-policer {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
    }
    filter limit-all-hosts {
        term one {
            from {
                source-address {
                    10.10.10/24;
                    11.11/16;
                }
            }
            then prefix-action per-source-policer;
        }
    }
}
```

Examples: Classify Traffic

Classify expedited forwarding traffic:

```
[edit]
firewall {
    policer ef-policer {
        if-exceeding {
            bandwidth-limit 300k;
            burst-size-limit 50k;
        }
        then {
            discard;
        }
    }
    term ef-multifield {
        then {
            loss-priority low;
            forwarding-class expedited-forwarding;
            policer ef-policer;
        }
    }
}
```

Classify assured forwarding traffic:

```
firewall {
    policer af-policer {
        if-exceeding {
            bandwidth-limit 300k;
            burst-size-limit 500k;
        }
        then {
            loss-priority high;
        }
    }
```

```
    term af-multifield {
        then {
            loss-priority low;
            forwarding-class assured-forwarding;
            policer af-policer;
        }
    }
}
```

Apply an Interface Policer

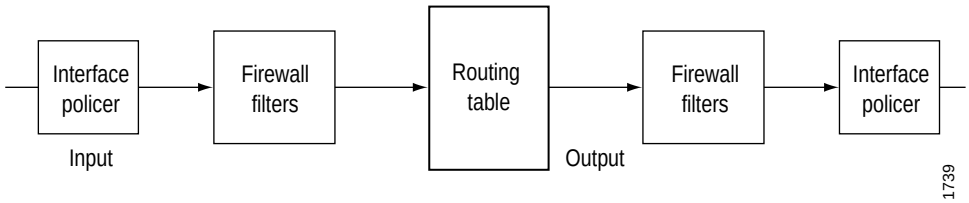
In addition to including policers in firewall filters, you can apply an interface policer that is not part of a firewall filter configuration. An interface policer can be applied to each family on an interface.

To apply an interface policer, include the policer statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name]
policer {
    input policer-name;
    output policer-name;
}
```

You must first configure the policer at the [edit firewall] hierarchy level before you can apply it to an interface. Both input and output policers are allowed, and can be used in conjunction with existing firewall filters. Input interface policers are evaluated before any input firewall filters. Likewise, output interface policers are evaluated after any output firewall filters (see Figure 12.)

Figure 12: Incoming and Outgoing Interface Policers



To display a policer on a particular interface, issue the show interfaces policers command at the command-line interface (CLI).

Example: Apply an Interface Policer

Apply a policer on circuit cross-connect (CCC) interfaces:

```
[edit interfaces]
so-0/0/0 {
    encapsulation ppp-ccc;
    unit 0 {
        family ccc {
            policer {
                input dragnet;
            }
        }
    }
}
```

Examples: Configure Policing

The following example shows a complete filter configuration containing a policer. It limits all FTP traffic from a given source to certain rate limits. Traffic exceeding the limits is discarded, and the remaining traffic is accepted and counted.

```
[edit]
firewall {
    policer policer-1 {
        if-exceeding {
            bandwidth-limit 400k;
            burst-size-limit 100k;
        }
        then {
            discard;
        }
    }
    term tcp-ftp {
        from {
            source-address 1.2.3/24;
            protocol tcp;
            destination-port ftp;
        }
        then {
            policer policer-1;
            accept;
            count count-ftp;
        }
    }
}
```

The following example shows a complete filter configuration containing two policers, and includes the next term action. Policer policer-1 limits all traffic from a given source to certain rate limits, then sets the forwarding class. Policer policer-2 limits all traffic to a second set of rate limits. Traffic exceeding the limits is discarded; the remaining traffic is accepted.

```
[edit]
firewall {
    policer policer-1 {
        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 100k;
        }
        then {
            forwarding-class 0;
        }
    }
    policer policer-2 {
        if-exceeding {
            bandwidth-limit 100m;
            burst-size-limit 100k;
        }
        then {
            discard;
        }
    }
    filter f {
        term term-1 {
            then {
                policer policer-1;
                next term;
            }
        }
        term term-2 {
            then {
                policer policer-2;
                accept;
            }
        }
    }
```

The following example limits all FTP traffic from a given source to certain rate limits, but defines the policer outside the filter, thereby creating a template that can be referenced by more than one filter or more than one term within a filter. Traffic exceeding the limits is discarded, and the remaining traffic is accepted and counted.

```
[edit]
firewall {
    policer policer-1 {
        if-exceeding {
            bandwidth-limit 400k;
            burst-size-limit 100k;
        }
        then {
            discard;
        }
    }
    filter limit-ftp {
        term tcp-ftp {
            from {
                source-address 1.2.3/24;
                protocol tcp;
                destination-port ftp;
            }
            then {
                policer policer-1;
                accept;
                count count-ftp;
            }
        }
    }
}
```

The following example shows a filter intended to thwart denial-of-service (DoS) SYN attacks:

```
[edit]
firewall {
    policer syn-recvd {
        if-exceeding {
            bandwidth-limit 40k;
            burst-size-limit 15000;
        }
        then discard;
    }
    term allow-syn {
        from {
            source-address {
                168.17.12.50/32;            # trusted addresses
            }
        }
        then {
            log;
            accept;
        }
    }
    term limit-syn {
        from {
            protocol tcp;
            tcp-initial;
        }
        then {
            count limit-syn;
            policer syn-recvd;
            accept;
        }
    }
    term default {
        then accept;
    }
}

[edit]                                  # apply filter to lo0 to control traffic to the Routing Engine
interfaces {
    lo0 {
        unit 0 {
            family inet {
                filter {
                input syn-attack;
                }
            }
            address 168.164.4.53/32;
        }
    }
}
```

The following example uses one filter to do the following:

- Stop all UDP and ICMP traffic destined to these addresses (in term a).
- Send ICMP through the policer (in term b).
- Accept ICMP traffic within contract and all other traffic (in term c).

It is important to keep the terms in order; once a packet has a match within the firewall filter, it is not examined in subsequent terms. For example, if you configured the filter to send ICMP traffic through the policer before discarding ICMP and UDP traffic to those addresses, it would not work.

```
[edit firewall]
policer policer-1 {
    if-exceeding {
        bandwidth-limit 200k;
        burst-size-limit 3k;
    }
    then {
        loss-priority 1;
        forwarding-class 1;
    }
}
term a {
    from {
        destination-address {
            104.126.50.2/23;
            188.130.12.1/23;
            163.82.16.0/24 except;
            163.82.0.3/18;
        }
        protocol [icmp udp];
    }
    then {
        count packets-dropped;
        discard;
    }
}
term b {
    from {
        protocol icmp;
    }
    then policer policer-1;
}
term c {
    then accept;
}
```

Chapter 12

Summary of Firewall Filter and Policer Configuration Statements

The following descriptions explain each of the firewall filter and policer configuration statements. The statements are organized alphabetically.

accounting-profile

Syntax accounting-profile *name*;

Hierarchy Level [edit firewall family *family-name* filter *filter-name*]

Description Enable collection of accounting data for the specified filter.

Options *name*—Name assigned to the accounting profile.

Usage Guidelines See “Configure a Firewall Filter Accounting Profile” on page 173.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

family

Syntax

```
family family-name {
    filter filter-name {
        accounting-profile name
        interface-specific
    }
    prefix-action name {
        count;
        destination-prefix-length prefix-length;
        policer policer-name;
        source-prefix-length prefix-length;
        subnet-prefix-length prefix-length;
    }
    prefix-policer {
        policer policer-name;
    }
}
```

Hierarchy Level [edit firewall]

Description Configure a firewall filter for IPv4 or IPv6 traffic.

Options *family-name*—Version of addressing protocol:

- inet—IPv4 addressing protocol.
- inet6—IPv6 addressing protocol.
- mpls—Multiprotocol Label Switching (MPLS) protocol.

The remaining statements are explained separately.

Usage Guidelines See “Configure the Family Address Type” on page 144.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

filter

Syntax

```
filter filter-name {
    accounting-profile name
    interface-specific
    term term-name {
        from {
            match-conditions;
        }
        then {
            action;
            action-modifiers;
        }
    }
}
```

Hierarchy Level [edit firewall family *family-name*]

Description Configure firewall filters.

Options *filter-name*—Name that identifies the filter. The name can contain letters, numbers, and hyphens (–) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

The remaining statements are explained separately.

Usage Guidelines See “Firewall Filter Configuration” on page 141.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

filter-specific

Syntax

```
filter-specific;
```

Hierarchy Level [edit firewall policer *policer-name*]

Description Configure a policer to act as a filter-specific policer. If this statement is not specified, then the policer defaults to a term-specific policer.

Usage Guidelines See “Configure Filter-Specific Policers” on page 189.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

firewall

Syntax firewall { … }

Hierarchy Level [edit]

Description Configure firewall filters.

The statements are explained separately.

Usage Guidelines See “Firewall Filter Configuration” on page 141.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

if-exceeding

Syntax
```
if-exceeding {
    bandwidth-limit bps;
    bandwidth-percent number;
    burst-size-limit bytes;
}
```

Hierarchy Level [edit firewall policer *policer-name*]

Description Configure policer rate limits.

Options bandwidth-limit *bps*—Traffic rate, in bits per second (bps). There is no minimum value, but any value below 61,040 bps results in an effective rate of 30,520 bps.
Range: 0 through 4.29 Gbps
Default: None

bandwidth-percent *number*—Port speed, in decimal percentage number.
Range: 1 through 100
Default: None

burst-size-limit *bytes*—Maximum burst size, in bytes. The minimum recommended value is the maximum transmission unit (MTU) of the IP packets being policed.
Range: 0 through 100 MB
Default: None

Usage Guidelines See “Configure Rate Limiting” on page 187.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

interface-specific

Syntax interface-specific;

Hierarchy Level [edit firewall family *family-name* filter *filter-name*]

Description Configure interface-specific names for firewall counters.

Usage Guidelines See “Configure Interface-Specific Counters” on page 169.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

policer

Syntax
```
policer policer-name {
   filter-specific;
   if-exceeding {
      bandwidth-limit bps;
      bandwidth-percent number;
      burst-size-limit bytes;
   }
   then {
      policer-action;
   }
}
```

Hierarchy Level [edit firewall]

Description Configure policer rate limits and actions. When included at the [edit firewall] hierarchy level, it creates a template, and you do not have to configure a policer individually for every firewall filter or interface. To activate a policer, you must include the policer action modifier in the then statement in a firewall filter term or on an interface.

Options *policer-action*—One or more actions to take:

- discard—Discard traffic that exceeds the rate limits.
- forwarding-class *class-name*—Specify the particular forwarding class.
- loss-priority—Set the packet loss priority (PLP) to low or high.

policer-name—Name that identifies the policer. The name can contain letters, numbers, and hyphens (–), and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

then—Actions to take on matching packets.

The remaining statements are explained separately.

Usage Guidelines See “Configure Policers” on page 187.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

prefix-action

Syntax
```
prefix-action name{
    count;
    destination-prefix-length prefix-length;
    policer policer-name;
    source-prefix-length prefix-length;
    subnet-prefix-length prefix-length;
}
```

Hierarchy Level [edit firewall family inet]

Description Configure prefix-specific action.

Options count—Enable counter.

destination-prefix-length *prefix-length*—Destination prefix length.
Range: 0 through 32

policer *policer-name*—Policer name.

source-prefix-length *prefix-length*—Source prefix length.
Range: 0 through 32

subnet-prefix-length *prefix-length*—Subnet prefix length.
Range: 0 through 32

Usage Guidelines See "Configure Prefix-Specific Action" on page 190.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

prefix-policer

Syntax
```
prefix-policer {
    policer policer-name;
}
```

Hierarchy Level [edit firewall family inet]

Description Apply a filter-specific policer.

Options *policer-name*—Name of the filter-specific policer with the filter-specific statement at the [edit firewall policer *policer-name*] hierarchy level.

Usage Guidelines See "Configure Filter-Specific Policers" on page 189.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

term

Syntax

```
term term-name {
   from {
      match-conditions;
   }
   then {
      actions;
      action-modifiers;
   }
}
```

Hierarchy Level [edit firewall family *family-name* filter *filter-name*]

Description Define a firewall filter term.

Options

actions—(Optional) An action to take if conditions match. If you do not specify an action, the packets that match the conditions in the from statement are accepted. The actions are described in Table 22 on page 147.

action-modifiers—(Optional) One or more actions to perform on a packet. The action modifiers are described in Table 22 on page 147.

from—(Optional) Match packet fields to values. If not included, all packets are considered to match and the actions and action modifiers in the then statement are taken.

match-conditions—One or more conditions to use to make a match. The conditions are described in Table 23 on page 152, Table 24 on page 153, Table 25 on page 156, and Table 26 on page 160.

term-name—Name that identifies the term. The name can contain letters, numbers, and hyphens (–), and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

then—(Optional) Actions to take on matching packets. If not included and a packet matches all the conditions in the from statement, the packet is accepted.

Usage Guidelines See “Configure Firewall Filters” on page 143.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Part 4

Traffic Sampling and Forwarding

Chapter 13
Traffic Sampling and Forwarding Overview

Traffic sampling allows you to sample IP traffic based on particular input interfaces and various fields in the packet header. You can also use traffic sampling to monitor any combination of specific logical interfaces, specific protocols on one or more interfaces, a range of addresses on a logical interface, or individual IP addresses. Information about the sampled packets is saved to files on the router's hard disk.

The forwarding policies allow you to configure the router for per-flow load balancing, port mirroring, and Domain Name System (DNS) or Trivial File Transfer Protocol (TFTP) forwarding.

Traffic sampling and forwarding require a router equipped with an Internet Processor II ASIC. To determine whether a router has an Internet Processor II ASIC, use the show chassis hardware command.

Traffic sampling is not meant to capture all packets received by a router. We do not recommend excessive sampling (a rate greater than 1/1000 packets), because it can increase the load on your processor. If you need to set a higher sampling rate to diagnose a particular problem or type of traffic received, we recommend that you revert to a lower sampling rate after you discover the problem or troublesome traffic.

Chapter 14

Traffic Sampling and Forwarding Configuration

To configure forwarding options and traffic sampling, include statements at the [edit forwarding-options] hierarchy level:

```
[edit forwarding-options]
family (inet | inet6 | mpls) {
    filter {
        input filter-name;
    }
}
hash-key {
    family inet {
        layer-3;
        layer-4;
    }
    family mpls {
        label-1;
        label-2;
    }
}
helpers {
    bootp {
        description description-of-service;
        interface interface-group {
            description description-of-interface;
            maximum-hop-count number;
            minimum-wait-time seconds;
            no-listen;
            server [ addresses ]
        maximum-hop-count number;
        minimum-wait-time seconds;
        server address < [ routing-instance routing-instance-name ] >;
    domain {
        description description-of-service;
        server address;
        interface interface-name {
            description description-of-interface;
            no-listen;
            server address < [ routing-instance routing-instance-name ] >;
        }
    }
```

```
    tftp {
        description description-of-service;
        server address < [ routing-instance routing-instance-name ] >;
        interface interface-name {
            description description-of-interface;
            no-listen;
            server address;
        }
    }
    traceoptions {
        file filename {
            files number;
            size bytes;
        }
        flag flag;
        level level;
    }
}
sampling {
    disable;
    input {
        family inet {
            max-packets-per-second number;
            rate number;
            run-length number;
        }
    }
    output {
        cflowd host-name {
            aggregation {
                autonomous-system;
                destination-prefix;
                protocol-port;
                source-destination-prefix {
                    caida-compliant;
                }
                source-prefix;
            }
            autonomous-system-type (origin | peer);
            (local-dump | no-local-dump);
            port port-number;
            version format;
        }
```

```
        file {
            disable;
            filename filename;
            files number;
            size bytes;
            (stamp | no-stamp);
            (world-readable | no-world-readable);
        }
        port-mirroring {
            interface interface-name {
                next-hop address;
            }
        }
    }
    traceoptions {
        file filename {
            files number;
            size bytes;
            (world-readable | no-world-readable);
        }
    }
}
```

This chapter describes the following tasks for configuring traffic sampling and forwarding options:

Minimum Traffic Sampling or Forwarding Configuration

To configure traffic sampling, you must perform at least the following tasks:

- Create a firewall filter to apply to the logical interfaces being sampled by including the filter statement at the [edit firewall family *family-name*] hierarchy level. In the filter then statement, you must specify the action modifier sample and the action accept.

```
[edit firewall family family-name]
filter filter-name {
    term term-name {
        then {
            sample;
            accept;
        }
    }
}
```

- Apply the filter to the interfaces on which you want to sample traffic:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family family-name {
            filter {
                input filter-name;
            }
            address address {
                destination destination-address;
            }
        }
    }
}
```

- Enable sampling and specify a nonzero sampling rate:

```
[edit forwarding-options]
sampling {
    input {
        family inet {
            rate number;
        }
    }
}
```

Configure a Forwarding Table Filter

A forwarding table filter allows you to filter data packets based on their components and to perform an action on packets that match the filter.

To apply a forwarding table filter to a forwarding table, include the filter input statement at the [edit forwarding-options family *family*] hierarchy level:

```
[edit]
forwarding-options {
    family family {
        filter {
            input filter-name;
        }
    }
}
```

Configure Traffic Sampling

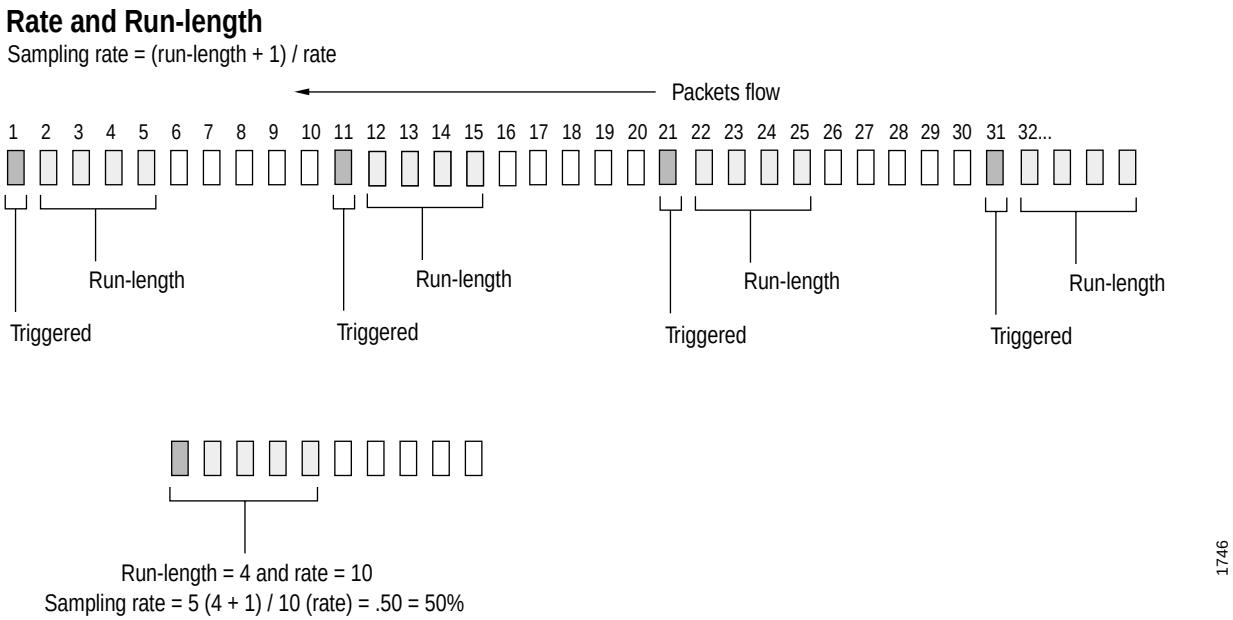
To configure traffic sampling on a logical interface, include the input statement at the [edit forwarding-options sampling] hierarchy level:

```
[edit forwarding-options sampling]
input {
    family inet {
        max-packets-per-second number;
        rate number;
        run-length number;
    }
}
```

Specify the threshold traffic value by using the max-packets-per-second statement. The value is the maximum number of packets to be sampled, beyond which the sampling mechanism begins dropping packets. The range is 0 through 65,535. A value of 0 instructs the Packet Forwarding Engine not to sample any packets. The default value is 1000.

Specify the sampling rate by setting the values for rate and run-length (see Figure 13).

Figure 13: Configure Sampling Rate



The rate statement specifies the ratio of packets to be sampled. For example, if you configure a rate of 10, *x* number of packets out of every 10 is sampled, where *x* = *run-length* + 1. By default, the rate is 0, which means that no traffic is sampled.

The run-length statement specifies the number of matching packets to sample following the initial one-packet trigger event. By default, the run-length is 0, which means that no more traffic is sampled after the trigger event. The range is 0 through 20. Configuring a run length greater than 0 allows you to sample packets following those already being sampled.

If you do not include the input statement, sampling is disabled.

To collect the sampled packets in a file, include the file statement at the [edit forwarding-options sampling output] hierarchy level. For more information about the output file formats, see "Specify the Files to Contain Traffic Sampling Output" on page 225.

Configure Per-Flow Load-Balancing Information

By default, when there are multiple equal-cost paths to the same destination, the JUNOS software chooses one of the next-hop addresses at random. On routers with the Internet Processor II ASIC, you have two additional options. You can specify what information the router uses for per-flow load balancing based on port data (instead of on source and destination IP addresses only). For aggregated Ethernet and aggregated SONET interfaces, you can load-balance based on the Multiprotocol Label Switching (MPLS) label information. For more information, see "Configure Load-Balance Per-Packet Action" on page 125.

Configure the Router or Interface to Act as a DHCP/BOOTP Relay Agent

You can configure the router or an interface to act as a Dynamic Host Configuration Protocol (DHCP) or Bootstrap Protocol (BOOTP) relay agent. This means that a locally attached host can issue a DHCP or BOOTP request as a broadcast message. If the router or an interface sees this broadcast message, it relays the message to a specified DHCP or BOOTP server.

You should configure the router or an interface to be a DHCP/BOOTP relay agent if you have locally attached hosts and a distant DHCP or BOOTP server.

This configuration overrides the DHCP or BOOTP configuration at the [edit system dhcp-relay] hierarchy level.

To configure the router to act as a DHCP/BOOTP relay agent, include the bootp statement at the [edit forwarding-options helpers] hierarchy level, specifying the address of the DHCP or BOOTP server:

```
[edit forwarding-options helpers]
bootp {
    description description-of-service;
    interface interface-group {
        description;
        maximum-hop-count number;
        minimum-wait-time seconds;
        no-listen;
        server [ addresses ];
    }
    maximum-hop-count number;
    minimum-wait-time seconds;
    server address < [ routing-instance routing-instance-names ] >;
```

description sets the description of the BOOTP service, DHCP service, or interface.

interface sets a logical interface or a group of logical interfaces with a specific DHCP-relay or BOOTP configuration.

routing-instance sets the routing instance of the server to forward. You can include as many routing instances as necessary in the same statement.

no-listen stops packets from being forwarded on a logical interface, a group of logical interfaces, or router.

maximum-hop-count sets the maximum allowed number in the hops field of the BOOTP header. Headers that have a larger number in the hops field are not forwarded. If you omit the maximum-hop-count statement, the default value is 4 hops.

minimum-wait-time sets the minimum allowed number of seconds in the secs field of the BOOTP header. Headers that have a smaller number in the secs field are not forwarded. If you omit the minimum-wait-time statement, the default value is 3 seconds.

server sets the IP address or addresses that specify the DHCP or BOOTP server for the router or interface. You can include as many addresses as necessary in the same statement.

You can also configure an individual logical interface to be a DHCP/BOOTP relay if you have locally attached hosts and a remote DHCP or BOOTP server at the [edit interfaces] hierarchy level. For more information, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service*.

Configure DNS and TFTP Packet Forwarding

You can configure the router to support DNS and TFTP packet forwarding for IPv4 traffic, which allows clients to send DNS or TFTP requests through the router. The responding DNS or TFTP server recognizes the client address and sends a response directly to that address. By default, the router ignores DNS and TFTP request packets.

To enable DNS or TFTP packet forwarding, include the helpers statement at the [edit forwarding-options] hierarchy level:

```
[edit forwarding-options]
helpers {
    domain {
        description description-of-service;
        server address < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description description-of-interface;
            no-listen;
            server address < [ routing-instance routing-instance-names ] >;
        }
    }
    tftp {
        description description-of-service;
        server address;
        interface interface-name {
            description description-of-interface;
            no-listen;
            server address < [ routing-instance routing-instance-names ] >;
        }
    }
}
```

domain sets domain packet forwarding.

description sets the description of the DNS or TFTP service.

tftp sets TFTP packet forwarding.

server sets a DNS or TFTP server (with an IPv4 address). Use one address for either a global configuration or for each interface.

routing-instance sets the routing instance of the server to forward. You can include as many routing instances as necessary in the same statement.

no-listen disables recognition of DNS or TFTP requests on one or more interfaces. If you do not specify at least one interface with this statement, the forwarding service is global to all interfaces on the router.

Trace BOOTP, DNS, and TFTP Forwarding Operations

Tracing operations track all traffic forwarding operations and record them in a log file in the /var/log directory. By default, this file is named /var/log/fud. The default file size is 128K, and 10 files are created before the first one gets overwritten.

To trace DNS and TFTP forwarding operations, include the traceoptions statement at the [edit forwarding-options helpers] hierarchy level:

```
[edit forwarding-options helpers]
traceoptions {
    file filename {
        files number;
        size bytes;
    }
    flag flag;
    level level;
}
```

Example: Configure DNS Packet Forwarding

Enable DNS packet request forwarding to all interfaces on the router except t1-1/1/2 and t1-1/1/3:

```
[edit forwarding-options helpers]
dns {
    server 10.10.10.30;
    interface {
        t1-1/1/2 {
            no-listen;
            server 10.10.10.9;
        }
        t1-1/1/3 {
            no-listen;
            server 10.10.10.4;
        }
}
```

Disable Traffic Sampling

To explicitly disable traffic sampling on the router, include the disable statement at the [edit forwarding-options] hierarchy level:

```
[edit forwarding-options]
sampling {
    disable;
}
```

Examples: Configure Traffic Sampling

The following sections provide examples of configuring traffic sampling:

Sample a Single SONET Interface

The following configuration gathers statistical sampling information from a small percentage of all traffic on a single SONET interface and collects it in a file named sonet-samples.txt.

Create the filter:

```
[edit firewall family inet]
filter {
    input sample-sonet {
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the SONET interface:

```
[edit interfaces]
so-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input sample-sonet;
            }
            address 216.127.68.254/32 {
                destination 216.127.74.7;
            }
        }
    }
}
```

Finally, configure traffic sampling:

```
[edit forwarding-options]
sampling {
    input {
        family inet {
            rate 100;
            run-length 2;
        }
    }
    output {
        file {
            filename sonet-samples-txt;
            files 40;
            size 5m;
        }
    }
}
```

Sample All Traffic from a Single IP Address

The following configuration gathers statistical information about every packet entering the router on a specific Gigabit Ethernet port originating from a single source IP address of 168.45.92.31, and collects it in a file named samples-168-45-92-31.txt.

Create the filter:

```
[edit firewall family inet]
filter one-ip {
    term get-ip {
        from {
            source-address 168.45.92.31;
        }
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the Gigabit Ethernet interface:

```
[edit interfaces]
ge-4/1/1 {
    unit 0 {
        family inet {
            filter {
                input one-ip;
            }
            address 215.45.92.254;
        }
    }
}
```

Finally, gather statistics on all the candidate samples; in this case, gather all statistics:

```
[edit forwarding-options]
sampling {
    input {
        family inet {
            rate 1;
        }
    }
    output {
        file {
            filename samples-215-45-92-31.txt;
            files 100;
            size 100k;
        }
    }
}
```

Sample All FTP Traffic

The following configuration gathers statistical information about a moderate percentage of packets using the FTP data transfer protocol in the output path of a specific T3 interface, and collects the information in a file named t3-ftp-traffic.txt.

Create a filter:

```
[edit firewall family inet]
filter ftp-stats {
    term ftp-usage {
        from {
            destination-port [ftp ftp-data];
        }
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the T3 interface:

```
[edit interfaces]
t3-7/0/2 {
    unit 0 {
        family inet {
            filter {
                input ftp-stats;
            }
            address 141.35.78.254/32 {
                destination 141.35.78.4;
            }
        }
    }
}
```

Finally, gather statistics on 10 percent of the candidate samples:

```
[edit forwarding-options]
sampling {
    input {
        family inet {
            rate 10;
        }
    }
    output {
        file {
            filename t3-ftp-traffic.txt;
            files 50;
            size 1m;
        }
    }
}
```

Specify the Files to Contain Traffic Sampling Output

You save traffic sampling results to a file in the /var/tmp directory. To collect the sampled packets in a file, include the file statement at the [edit forwarding-options sampling output] hierarchy level:

```
[edit forwarding-options sampling output]
file {
    disable;
    filename filename;
    files number;
    size bytes;
    (stamp | no-stamp);
    (world-readable | no-world-readable);
}
```

Traffic Sampling Output Files

Traffic sampling output is saved to an ASCII text file. The following is an example of the traffic sampling output that is saved to a file in the /var/tmp directory. Each line in the output file contains information for one sampled packet. You can optionally display a timestamp for each line.

The column headers are repeated after each group of 1000 packets.

```
# Apr  7 15:48:50
Time                    Dest           Src Dest Src Proto TOS Pkt Intf  IP   TCP
                        addr          addr port port          len num frag flags
Apr 7 15:48:54 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:55 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:56 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:57 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:58 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
```

The output contains the following fields:

- Time—Time at which the packet was received (displayed only if you include the stamp statement in the configuration)
- Dest addr—Destination IP address in the packet
- Src addr—Source IP address in the packet
- Dest port—TCP or UDP port for the destination address
- Src port—TCP or UDP port for the source address
- Proto—Packet's protocol type
- TOS—Contents of the type-of-service (ToS) field in the IP header
- Pkt len—Length of the sampled packet, in bytes
- Intf num—Unique number that identifies the sampled logical interface
- IP frag—IP fragment number, if applicable
- TCP flags—Any TCP flags found in the IP header

To set the timestamp option for the file my-sample, enter the following:

```
[edit forwarding-options sampling output file]
user@host# set filename my-sample files 5 size 2m world-readable stamp;
```

Whenever you toggle the timestamp option, a new header is included in the file. If you set the stamp option, the Time field is displayed.

```
# Apr  7 15:48:50
# Time            Dest        Src  Dest   Src Proto  TOS   Pkt  Intf    IP   TCP
#                 addr       addr  port  port              len   num  frag flags
# Feb  1 20:31:21
#                 Dest        Src  Dest   Src Proto  TOS   Pkt  Intf    IP   TCP
#                 addr       addr  port  port              len   num  frag flags
```

Trace Traffic Sampling Operations

Tracing operations track all traffic sampling operations and record them in a log file in the /var/log directory. By default, this file is named /var/log/sampled. The default file size is 128K, and 10 files are created before the first one gets overwritten.

To trace traffic sampling operations, include the traceoptions statement at the [edit forwarding-options sampling] hierarchy level:

```
[edit forwarding-options sampling]
traceoptions {
    file filename {
        files number;
        size bytes;
        (world-readable | no-world-readable);
    }
}
```

Configure Flow Aggregation (cflowd)

You can collect an aggregate of sampled flows and send the aggregate to a specified host that runs the cflowd application available from CAIDA (http://www.caida.org). By using cflowd, you can obtain various types of byte and packet counts of flows through a router.

The cflowd application collects the sampled flows over a period of 1 minute. At the end of the minute, the number of samples to be exported are divided over the period of another minute and are exported over the course of the same minute.

Before you can perform flow aggregation, the routing protocol process must export the AS path and routing information to the sampling process. To do this, include the route-record statement at the [edit routing-options] hierarchy level (for routing instances, include the statement at the [edit routing-instances *routing-instance-name* routing-options] hierarchy level):

```
[edit]
routing-options {
    route-record;
}
```

By default, flow aggregation is disabled. To enable the collection of flow aggregates, include the cflowd statement at the [edit forwarding-options sampling output] hierarchy level:

```
[edit forwarding-options sampling output]
cflowd host-name {
    aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
            caida-compliant;
        }
        source-prefix;
    }
    autonomous-system-type (origin | peer);
    (local-dump | no-local-dump);
    port port-number;
    version format;
}
```

In the cflowd statement, specify the name or identifier of the host that collects the flow aggregates. You must also include the UDP port number on the host and the version, which gives the format of the exported cflowd aggregates. To collect cflowd records in a log file before exporting, include the local-dump statement.

You cannot specify both host (cflowd) sampling and port mirroring in the same configuration.

To specify aggregation of specific types of traffic, include the aggregation statement. This conserves memory and bandwidth enabling cflowd to export targeted flows rather than all the aggregated traffic. To specify a flow type, include the aggregation statement at the [edit forwarding-options sampling output cflowd *host-name*] hierarchy level:

```
[edit forwarding-options sampling output cflowd host-name]
aggregation {
    source-destination-prefix;
}
```

You specify the aggregation type using one of the following options:

- autonomous-system—Aggregate by autonomous system (AS) number; may require setting the separate cflowd autonomous-system-type statement to include either origin or peer AS numbers. The origin option specifies to use the origin AS of the packet source address in the Source Autonomous System cflowd field. The peer option specifies to use the peer AS through which the packet passed in the Source Autonomous System cflowd field. By default, cflowd exports the origin AS number.
- destination-prefix—Aggregate by destination prefix (only).
- protocol-port—Aggregate by protocol and port number; requires setting the separate cflowd port statement.

- source-destination-prefix—Aggregate by source and destination prefix. Version 2.1b1 of CAIDA's cflowd application does not record source and destination mask length values in compliance with CAIDA's *cflowd Configuration Guide*, dated August 30, 1999. If you configure the caida-compliant statement, the JUNOS software complies with Version 2.1b1 of cflowd. If you do not include the caida-compliant statement in the configuration, the JUNOS software records source and destination mask length values in compliance with the *cflowd Configuration Guide*.

- source-prefix—Aggregate by source prefix (only).

Collection of sampled packets in a local ASCII file is not affected by the cflowd statement.

Debug cflowd Flow Aggregation

To collect the cflowd flows in a log file before they are exported, include the local-dump option at the [edit forwarding-options sampling output cflowd *hostname*] hierarchy level:

```
[edit forwarding-options sampling output cflowd host-name]
local-dump;
```

By default, the flows are collected in /var/log/sampled; to change the filename, include the filename statement at the [edit forwarding-options sampling traceoptions] hierarchy level. For more information about changing the filename, see "Specify the Files to Contain Traffic Sampling Output" on page 225.

Because the local-dump option adds extra overhead, you should use it only while debugging cflowd problems, not during normal operation.

The following is an example of the flow information. The AS number exported is the origin AS number. All flows that belong under a cflowd header are dumped, followed by the header itself:

```
Jun 27 18:35:43 v5 flow entry
Jun 27 18:35:43    Src addr: 192.53.127.1
Jun 27 18:35:43    Dst addr: 192.6.255.15
Jun 27 18:35:43    Nhop addr: 192.6.255.240
Jun 27 18:35:43    Input interface: 5
Jun 27 18:35:43    Output interface: 3
Jun 27 18:35:43    Pkts in flow: 15
Jun 27 18:35:43    Bytes in flow: 600
Jun 27 18:35:43    Start time of flow: 7230
Jun 27 18:35:43    End time of flow: 7271
Jun 27 18:35:43    Src port: 26629
Jun 27 18:35:43    Dst port: 179
Jun 27 18:35:43    TCP flags: 0x10
Jun 27 18:35:43    IP proto num: 6
Jun 27 18:35:43    TOS: 0xc0
Jun 27 18:35:43    Src AS: 7018
Jun 27 18:35:43    Dst AS: 11111
Jun 27 18:35:43    Src netmask len: 16
Jun 27 18:35:43    Dst netmask len: 0
```

[... 41 more v5 flow entries; then the following header:]

```
Jun 27 18:35:43 cflowd header:
Jun 27 18:35:43   Num-records: 42
Jun 27 18:35:43   Version: 5
Jun 27 18:35:43   Flow seq num: 118
Jun 27 18:35:43   Engine id: 0
Jun 27 18:35:43   Engine type: 3
```

Configure Port Mirroring

On routers containing a Internet Processor II ASIC, you can send a copy of an IPv4 packet from the router to an external host address or a packet analyzer for analysis. This is known as *port mirroring*.

Port mirroring is different from traffic sampling. In traffic sampling, a sampling key based on the IPv4 header is sent to the Routing Engine. There, the key can be placed in a file, or cflowd packets based on the key can be sent to a cflowd server. In port mirroring, the entire packet is copied and sent out through a next-hop interface.

To configure port mirroring, include the statements described in “Configure a Forwarding Table Filter” on page 217. Specify the port-mirroring destination by including the next-hop statement at the [edit forwarding-options sampling output] hierarchy level:

```
[edit forwarding-options sampling output]
port-mirroring {
    interface interface-name {
        next-hop address;
    }
}
```

The interface is the output interface used to send the packets to the analyzer. You can use any physical interface type, including generic routing encapsulation (GRE) tunnel interfaces. The next-hop address specifies the destination address; this statement is mandatory for non-point-to-point interfaces, such as Ethernet interfaces.

To configure the sampling rate, include the rate statement at the [edit forwarding-options sampling input family inet] hierarchy level.

In typical applications, you send the sampled packets to an analyzer or a workstation for analysis, rather than another router. If you must send this traffic over a network, you should use tunnels. For more information about tunnel interfaces, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service.*

The following restrictions apply:

- You cannot configure both cflowd sampling and port mirroring in the same configuration.
- You cannot configure firewall filters on the port-mirroring interface.
- The interface you configure for port mirroring should not participate in any kind of routing activity.
- The destination address you specify should not have a route to the ultimate traffic destination. For example, if the sampled IPv4 packets have a destination address of 190.68.9.10 and the port-mirrored traffic is sent to 190.68.20.15 for analysis, the device associated with the latter address should not know a route to 190.68.9.10. Also, it should not send the sampled packets back to the source address.
- Only IPv4 traffic is supported.
- Only transit data is supported.
- You can configure only one port-mirroring interface per router. If you include more than one interface in the port-mirroring statement, the previous one is overwritten.
- You must include a firewall filter with both the accept action and the sample action modifier on the inbound interface. Do not include the discard action, or port mirroring will not work.

Examples: Configure Port Mirroring

To configure port mirroring, do the following:

- Enable sampling by configuring a firewall filter.
- Apply that firewall filter on the inbound and, optionally, the outbound interface.
- Include the port-mirroring statement in the configuration.

You do not need to configure firewall filters on both interfaces, but at least one is necessary on the inbound interface to provide the copies of the packets to send to an analyzer.

Following are some basic configuration examples.

Configure a firewall filter to be applied to interface so-3/2/1, logical interface 0 (inbound) and interface so-0/0/2, logical interface 0 (outbound):

```
[edit firewall]
filter sample-all {
    term default {
        then {
            sample;
            accept;
        }
    }
}
```

Configuration for the inbound interface:

```
[edit interfaces so-3/2/1]
sonet-options {
    fcs 32;
}
unit 0 {
    family inet {
        filter {
            input sample-all;
        }
        address 10.26.13.1/32 {
            destination 10.26.13.2;
        }
    }
}
```

Configuration for the outbound interface:

```
[edit interfaces so-0/0/2]
sonet-options {
    fcs 32;
}
unit 0 {
    family inet {
        filter {
            output sample-all;
        }
        address 10.36.11.1/32 {
            destination 10.36.11.2;
        }
    }
}
```

Configurations for the port-mirroring interface:

- When the port-mirroring interface is a non-point-to-point interface:

```
[edit interfaces ge-3/2/0]
unit 0 {
    family inet {
        address 10.36.7.1/24;
    }
}
```

- When the port-mirroring interface is a point-to-point interface:

```
[edit interfaces so-1/0/2]
sonet-options {
    fcs 32;
}
unit 0 {
    family inet {
        address 10.36.6.1/32 {
            destination 10.36.6.2;
        }
    }
}
```

If the port-mirroring interface is a non-point-to-point interface, you must include an IP address under the port-mirroring statement to identify the other end of the link. This IP address must be reachable for you to see the sampled traffic. If the port-mirroring interface is an Ethernet interface, the router should have an Address Resolution Protocol (ARP) entry for it. The following sample configuration sets up a static ARP entry:

```
[edit interfaces ge-3/2/0]
unit 0 {
    family inet {
        address 10.36.7.1/24 {
            arp 10.36.7.2 mac 00:90:69:4f:84:9d;
        }
    }
}
```

IP address 10.36.7.1 is the router address and 10.36.7.2 is the address at the remote end. The Media Access Control (MAC) address entry maps to 10.36.7.2. When you configure interface ge-3/2/0 as the port-mirroring interface, the router continues to sample even if you clear the ARP cache. For more information on ARP entries, see the *JUNOS Internet Software Configuration Guide: Interfaces and Class of Service* or the *JUNOS Internet Software Operational Mode Command Reference.*

The following configuration applies when the port-mirroring interface is a non-point-to-point interface:

```
[edit forwarding-options]
sampling {
    input {
        family inet {
            rate 1;
        }
    }
    output {
        port-mirroring {
            interface ge-3/2/0.0 {
                next-hop 10.36.7.2;
            }
        }
    }
}
```

If the port-mirroring interface is a point-to-point interface, you do not need to include an IP address under the port-mirroring statement:

```
[edit forwarding-options]
sampling {
    input {
        family inet {
            rate 1;
        }
    }
    output {
        port-mirroring {
            interface so-1/0/2;
        }
    }
}
```

Chapter 15

Summary of Traffic Sampling and Forwarding Options Configuration Statements

The following sections explain each of the sampling and forwarding statements. The statements are organized alphabetically.

aggregation

Syntax

```
aggregation {
    autonomous-system;
    destination-prefix;
    protocol-port;
    source-destination-prefix {
        caida-compliant;
    }
    source-prefix;
}
```

Hierarchy Level [edit forwarding-options sampling output cflowd *host-name*]

Description Specify the type of data to be aggregated; cflowd records and sends only those flows that match the specified criteria.

Options

autonomous-system—Aggregate by autonomous system (AS) number.

caida-compliant—Record source and destination mask length values in compliance with the Version 2.1b1 release of CAIDA's "cflowd" application. If this statement is not configured, the JUNOS software records source and destination mask length values in compliance with the *cflowd Configuration Guide,* dated August 30, 1999.

destination-prefix—Aggregate by destination prefix.

protocol-port—Aggregate by protocol and port number.

source-destination-prefix—Aggregate by source and destination prefix.

source-prefix—Aggregate by source prefix.

Usage Guidelines See "Configure Flow Aggregation (cflowd)" on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

autonomous-system-type

Syntax autonomous-system-type (origin | peer);

Hierarchy Level [edit forwarding-options sampling output cflowd *host-name*]

Description Specify the type of AS numbers that cflowd exports.

Options origin—Export origin AS numbers of the packet source address in the Source Autonomous System cflowd field.

peer—Export peer AS numbers through which the packet passed in the Source Autonomous System cflowd field.

Default origin

Usage Guidelines See “Configure Flow Aggregation (cflowd)” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

bootp

Syntax

```
bootp {
    description description-of-service;
    interface interface-group {
        description description-of-interface;
        maximum-hop-count;
        minimum-wait-time seconds;
        no-listen;
        server address < [ routing-instance routing-instance-names ] >;
    }
    maximum-hop-count;
    minimum-wait-time seconds;
    server [ addresses ];
}
```

Hierarchy Level [edit forwarding-options helpers]

Description Configures a router or interface to act as a Dynamic Host Configuration Protocol (DHCP) or (BOOTP) relay agent.

DHCP relaying is disabled.

Options description—Description of DHCP or BOOTP service.

no-listen—Stops packets from being forwarded on a logical interface, a group of logical interfaces, or router.

maximum-hop-count *number*—In the hops field of the BOOTP header, the maximum number of hops allowed.
Default: 4 hops

minimum-wait-time *seconds*—In the secs field of the BOOTP header, the minimum time allowed.
Default: 3 seconds

server [*addresses*]—Sets the IP address or addresses that specify the DHCP server or BOOTP server for the router or interface.

routing-instance *routing-instance-name*—The routing instance of the server to forward. You can include as many routing instances as necessary in the same statement.

interface *interface-group*—Sets a logical interface or group of logical interfaces with a specific DHCP relay configuration.

Usage Guidelines See “Configure Flow Aggregation (cflowd)” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

cflowd

Syntax

```
cflowd host-name {
    aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
            caida-compliant;
        }
        source-prefix;
    }
    autonomous-system-type (origin | peer);
    (local-dump | no-local-dump);
    port port-number;
    version format;
}
```

Hierarchy Level [edit forwarding-options sampling output]

Description Collect an aggregate of sampled flows and send the aggregate to a specified host system that runs the collection utility cfdcollect.

Options *host-name*—The IP address or identifier of the host system (the workstation running the cflowd utility).

The remaining statements are explained separately.

Usage Guidelines See “Configure Flow Aggregation (cflowd)” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

description

description (interface)

Syntax description *description-of-interface*;

Hierarchy Level [edit forwarding-options helpers bootp interface *interface-name*]
[edit forwarding-options helpers domain interface *interface-name*]
[edit forwarding-options helpers tftp interface *interface-name*]

Description Text description of interface.

The remaining statements are explained separately.

Usage Guidelines See “Configure DNS and TFTP Packet Forwarding” on page 220.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

description (service)

Syntax description *description-of-service*;

Hierarchy Level [edit forwarding-options helpers bootp]
[edit forwarding-options helpers domain]
[edit forwarding-options helpers tftp]

Description Description of BOOTP, DHCP, DNS, or TFTP service.

The remaining statements are explained separately.

Usage Guidelines See "Configure the Router or Interface to Act as a DHCP/BOOTP Relay Agent" on page 219 and "Configure DNS and TFTP Packet Forwarding" on page 220.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

disable

Syntax disable;

Hierarchy Level [edit forwarding-options sampling],
[edit forwarding-options sampling file]

Description Disable traffic sampling.

Usage Guidelines See "Disable Traffic Sampling" on page 221.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

domain

Syntax
```
domain {
    description description-of-service;
    server address < [ routing-instance routing-instance-name ] >;
    interface interface-name {
        description description-of-interface;
        no-listen;
        server address < [ routing-instance routing-instance-name ] >;
    }
}
```

Hierarchy Level [edit forwarding-options helpers]

Description Enable DNS request packet forwarding.

Options The statements are explained separately.

Usage Guidelines See "Configure DNS and TFTP Packet Forwarding" on page 220.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

family inet

family inet (for load balancing)

Syntax
```
family inet {
    layer-3;
    layer-4;
}
```

Hierarchy Level [edit forwarding-options hash-key]

Description Configure layer information for the load balancing specification. Only the IPv4 protocol is supported.

Options layer-3—Include Layer 3 (IP) data in the hash key.

layer-4—Include Layer 4 TCP or UDP data in the hash key.

Usage Guidelines See “Configure Per-Flow Load-Balancing Information” on page 218.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

family inet (for sampling rates)

Syntax
```
family inet {
    max-packets-per-second number;
    rate number;
    run-length number;
}
```

Hierarchy Level [edit forwarding-options sampling input]

Description Configure the protocol family to be sampled. Only the IPv4 protocol is supported.

Options The statements are explained separately.

Usage Guidelines See “Configure a Forwarding Table Filter” on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

family mpls

Syntax
```
family mpls {
    label-1;
    label-2;
}
```

Hierarchy Level [edit forwarding-options hash-key]

Description For aggregated Ethernet and SONET interfaces only, configure load balancing based on Multiprotocol Label Switching (MPLS) labels. Only the IPv4 protocol is supported.

Options label-1—Include only one label in the hash key.

label-2—Include both labels in the hash key.

Usage Guidelines See “Configure Per-Flow Load-Balancing Information” on page 218.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

file

file (Collect Traffic Samples)

Syntax
```
file {
    disable;
    filename filename;
    files number;
    size bytes;
    (stamp | no-stamp);
    (world-readable | no-world-readable);
}
```

Hierarchy Level [edit forwarding-options sampling output]

Description Collect the traffic samples in a file.

The statements are explained separately.

Usage Guidelines See “Specify the Files to Contain Traffic Sampling Output” on page 225.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

file (collect trace information)

Syntax
```
file filename {
    files number;
    size bytes;
    (world-readable | no-world-readable);
}
```

Hierarchy Level [edit forwarding-options sampling traceoptions]

Description Configure information about the files that contain trace logging information.

Options *filename*—The name of the file containing the trace information.
Default: /var/log/sampled

The remaining statements are explained separately.

Usage Guidelines See "Trace Traffic Sampling Operations" on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

file (for helpers traceoptions)

Syntax
```
file filename {
    files number;
    size bytes;
}
```

Hierarchy Level [edit forwarding-options helpers traceoptions]

Description Configure information about the DNS and TFTP packet-forwarding files that contain trace logging information.

Options *filename*—Name of the file containing the trace information.
Default: /var/log/sampled

flag *flag*—Name of the flag.

The remaining statements are explained separately.

Usage Guidelines See "Trace Traffic Sampling Operations" on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

filename

Syntax filename *filename*;

Hierarchy Level [edit forwarding-options sampling output file]

Description Configure the name of the output file.

Options *filename*—Name of the file in which to place the traffic samples. All files are placed in the directory /var/tmp.

Usage Guidelines See "Specify the Files to Contain Traffic Sampling Output" on page 225.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

files

Syntax files *number*;

Hierarchy Level [edit forwarding-options sampling output file],
[edit forwarding-options sampling traceoptions file]

Description Configure the total number of files to be saved with samples or trace data.

Options *number*—Maximum number of traffic sampling or trace log files. When a file named *sampling-file* reaches its maximum size, it is renamed *sampling-file*.0, then *sampling-file*.1, and so on, until the maximum number of traffic sampling files is reached. Then the oldest sampling file is overwritten.
Range: 1 through 100 files
Default: 5 files for sampling output; 10 files for trace log information

Usage Guidelines See "Specify the Files to Contain Traffic Sampling Output" on page 225 and "Trace Traffic Sampling Operations" on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

filter

Syntax filter input *filter-name*;

Hierarchy Level [edit forwarding-options family *family*]

Description Apply a forwarding table filter to a forwarding table.

Options input *filter-name*—Name of the forwarding table filter.

Usage Guidelines See "Configure a Forwarding Table Filter" on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

forwarding-options

Syntax forwarding-options { ... }

Hierarchy Level [edit]

Description Configure traffic forwarding.

The statements are explained separately.

Usage Guidelines See “Traffic Sampling and Forwarding Configuration” on page 213.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

hash-key

Syntax
```
hash-key {
    family inet {
        layer-3;
        layer-4;
    }
    family mpls {
        label-1;
        label-2;
    }
}
```

Hierarchy Level [edit forwarding-options]

Description Select which packet header data to use for per-flow load balancing.

The statements are explained separately.

Usage Guidelines See “Configure Per-Flow Load-Balancing Information” on page 218.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

helpers

Syntax

```
helpers {
    bootp {
        description description-of-service;
        interface interface-group {
            description description-of-interface;
            maximum-hop-count number;
            minimum-wait-time seconds;
            no-listen;
            server [ addresses ];
        }
        maximum-hop-count number;
        minimum-wait-time seconds;
        server address < [ routing-instance routing-instance-names ] >;
    }
    domain {
        description description-of-service;
        server address < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description description-of-interface;
            no-listen;
            server address < [ routing-instance routing-instance-names ] >;
        }
    }
    tftp {
        description description-of-service;
        server address < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description description-of-interface;
            no-listen;
            server address < [ routing-instance routing-instance-names ] >;
        }
    }
    traceoptions {
        file filename {
            files number;
            size bytes;
        }
        level level;
        flag flag;
    }
}
```

Hierarchy Level [edit forwarding-options]

Description Enable TFTP or DNS request packet forwarding, or configure the router or interface to act as a DHCP/BOOTP relay agent. Use only one server address per interface or global configuration.

Options The statements are explained separately.

Usage Guidelines See “Configure DNS and TFTP Packet Forwarding” on page 220 and “Configure the Router or Interface to Act as a DHCP/BOOTP Relay Agent” on page 219.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

input

Syntax
```
input {
    family inet {
        max-packets-per-second number;
        rate number;
        run-length number;
    }
}
```

Hierarchy Level [edit forwarding-options sampling]

Description Configure traffic sampling on a logical interface.

The statements are explained separately.

Usage Guidelines See “Configure a Forwarding Table Filter” on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

interface

interface (for port mirroring)

Syntax interface *interface-name*;

Hierarchy Level [edit forwarding-options sampling output port-mirroring]

Description Specify the output interface for sending copies of packets elsewhere to be analyzed.

Options *interface-name*—Name of the interface.

Usage Guidelines See “Configure Port Mirroring” on page 230.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

interface (for DNS and TFTP packet forwarding or relay agent)

Syntax interface *interface-name*;

Hierarchy Level [edit forwarding-options helpers bootp]
[edit forwarding-options helpers domain]
[edit forwarding-options helpers tftp]

Description Specify the interface for monitoring and forwarding DNS or TFTP requests, or specify an interface for a DHCP or BOOTP relay agent.

Options *interface-name*—Name of the interface.

Usage Guidelines See "Configure DNS and TFTP Packet Forwarding" on page 220 and "Configure the Router or Interface to Act as a DHCP/BOOTP Relay Agent" on page 219.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

local-dump

Syntax (local-dump | no-local-dump);

Hierarchy Level [edit forwarding-options sampling output cflowd *host-name*]

Description Enable collection of cflowd records in a log file.

Options no-local-dump—Do not dump cflowd records to a log file before exporting.

local-dump—Dump cflowd records to a log file before exporting.

Usage Guidelines See "Debug cflowd Flow Aggregation" on page 229.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

max-packets-per-second

Syntax max-packets-per-second *number* ;

Hierarchy Level [edit forwarding-options sampling input family inet]

Description Specify the traffic threshold that must be exceeded before packets are dropped. A value of 0 instructs the Packet Forwarding Engine not to sample any traffic.

Options *number*— Maximum number of packets per second.
Range: 0 through 65,535
Default: 1000

Usage Guidelines See "Configure a Forwarding Table Filter" on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

next-hop

Syntax next-hop *address*;

Hierarchy Level [edit forwarding-options sampling output port-mirroring]

Description Specify the next-hop address for sending copies of packets to an analyzer.

Options *address*—IP address of the next-hop router.

Usage Guidelines See “Configure Port Mirroring” on page 230.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

no-listen

Syntax no-listen;

Hierarchy Level [edit forwarding-options helpers bootp interface *interface-name*]
[edit forwarding-options helpers domain interface *interface-name*]
[edit forwarding-options helpers tftp interface *interface-name*]

Description Disable recognition of DNS requests or stop packets from being forwarded on a logical interface, a group of logical interfaces, or a router.

Usage Guidelines See “Configure DNS and TFTP Packet Forwarding” on page 220 and “Configure the Router or Interface to Act as a DHCP/BOOTP Relay Agent” on page 219.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

no-local-dump

See local-dump on page 247

no-stamp

See stamp on page 254

no-world-readable

See world-readable on page 258

output

Syntax

```
output {
    cflowd host-name {
        aggregation {
            autonomous-system;
            destination-prefix;
            protocol-port;
            source-destination-prefix {
                caida-compliant;
            }
            source-prefix;
        }
        autonomous-system-type (origin | peer);
        (local-dump | no-local-dump);
        port port-number;
        version format;
    }
    file {
        filename;
        files number;
        size bytes;
        (stamp | no-stamp);
        (world-readable | no-world-readable);
    }
    port-mirroring {
        interface interface-name;
        next-hop address;
    }
}
```

Hierarchy Level [edit forwarding-options sampling]

Description Configure cflowd, output files, and port mirroring.

The statements are explained separately.

Usage Guidelines See “Configure Flow Aggregation (cflowd)” on page 227, “Specify the Files to Contain Traffic Sampling Output” on page 225, and “Configure Port Mirroring” on page 230.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

port

Syntax port *port-number*;

Hierarchy Level [edit forwarding-options sampling output cflowd *host-name*]

Description Specify the UDP port number on the cflowd host system.

Options *port-number*—Any valid UDP port number on the host system.

Usage Guidelines See “Configure Flow Aggregation (cflowd)” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

port-mirroring

Syntax
```
port-mirroring {
    interface interface-name;
    next-hop address;
}
```

Hierarchy Level [edit forwarding-options sampling output]

Description Specify the interface and next-hop address for sending copies of packets to an analyzer.

The remaining statements are explained separately.

Usage Guidelines See “Configure Port Mirroring” on page 230.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

rate

Syntax rate *number* ;

Hierarchy Level [edit forwarding-options sampling input]

Description Set the ratio of the number of packets to be sampled. For example, if you specify a rate of 10, every tenth packet (1 packet out of 10) is sampled.

Options *number*—Denominator of the ratio.
Range: 1 through 65,535

Usage Guidelines See “Configure a Forwarding Table Filter” on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

routing-instance

Syntax routing-instance [*routing-instance-names*];

Hierarchy Level [edit forwarding-options helpers bootp server];
[edit forwarding-options helpers domain server];
[edit forwarding-options helpers domain server interface *interface-name*];
[edit forwarding-options helpers tftp server];
[edit forwarding-options helpers tftp server interface *interface-name*]

Description Set the routing instance of the server to forward. You can include as many routing instances as necessary in the same statement.

Options *routing-instance-names*—Name of the routing instance.

Usage Guidelines See “Configure a Forwarding Table Filter” on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

run-length

Syntax run-length *number* ;

Hierarchy Level [edit forwarding-options sampling input family inet]

Description Set the number of samples following the initial trigger event. This allows you to sample packets following those already being sampled.

Options *number*—Number of samples.
Range: 0 through 20
Default: 0

Usage Guidelines See “Configure a Forwarding Table Filter” on page 217.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

sampling

Syntax

```
sampling {
    disable;
    input {
        family inet {
            max-packets-per-second number;
            rate number;
            run-length number;
        }
    }
    output {
        cflowd host-name {
            aggregation {
                autonomous-system;
                destination-prefix;
                protocol-port;
                source-destination-prefix {
                    caida-compliant;
                }
                source-prefix;
            }
            autonomous-system-type (origin | peer);
            (local-dump | no-local-dump);
            port port-number;
            version format;
        }
        file {
            disable;
            filename filename;
            files number;
            size bytes;
            (stamp | no-stamp);
            (world-readable | no-world-readable);
        }
        port-mirroring {
            interface interface-name;
            next-hop address;
        }
    }
    traceoptions {
        file filename {
            files number;
            size bytes;
            (world-readable | no-world-readable);
        }
    }
}
```

Hierarchy Level [edit forwarding-options]

Description Configure traffic sampling.

The statements are explained separately.

Usage Guidelines See “Configure a Forwarding Table Filter” on page 217, “Configure Flow Aggregation (cflowd)” on page 227, “Configure Port Mirroring” on page 230, and “Trace Traffic Sampling Operations” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

server

server (DNS and TFTP service)

Syntax server *address*;

Hierarchy Level
[edit forwarding-options helpers domain interface *interface-name*],
[edit forwarding-options helpers tftp interface *interface-name*],
[edit forwarding-options helpers domain]

Description Specify the DNS or TFTP server for forwarding DNS or TFTP requests. Only one server can be specified for each interface.

Options *address*—Address of the server.

Usage Guidelines See “Configure DNS and TFTP Packet Forwarding” on page 220.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

server (DHCP or BOOTP service)

Syntax server [*addresses*];

Hierarchy Level [edit forwarding-options helpers bootp],
[edit forwarding-options helpers bootp interface *interface-group*],

Description Configure the router to act as a DHCP or BOOTP relay agent.

Options [*addresses*]—One or more addresses of the server.

Usage Guidelines See “Configure the Router or Interface to Act as a DHCP/BOOTP Relay Agent” on page 219.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

size

Syntax size *bytes*;

Hierarchy Level [edit forwarding-options sampling output file],
[edit forwarding-options sampling traceoptions file]

Description Specify the maximum size of each file containing sample or log data. The file size is limited by the number of files to be created and the available hard disk space.

When a traffic sampling file named *sampling-file* reaches the maximum size, it is renamed *sampling-file.*0. When the *sampling-file* again reaches its maximum size, *sampling-file.*0 is renamed *sampling-file.*1 and *sampling-file* is renamed *sampling-file.*0. This renaming scheme continues until the maximum number of traffic sampling files is reached. Then the oldest traffic sampling file is overwritten.

Options *bytes*—Maximum size of each traffic sampling file or trace log file, in kilobytes (KB), megabytes (MB), or gigabytes (GB).
Syntax: *x*k to specify KB, *x*m to specify MB, or *x*g to specify GB
Range: 10 KB through the maximum file size supported on your router
Default: 1 MB for sampling data; 128 KB for log information

Usage Guidelines See “Specify the Files to Contain Traffic Sampling Output” on page 225 and “Trace Traffic Sampling Operations” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

stamp

Syntax (stamp | no-stamp);

Hierarchy Level [edit forwarding-options sampling output file]

Description Include a timestamp with each line in the output file.

Options no-stamp—Do not include timestamps. This is the default.

stamp—Include a timestamp with each line of packet sampling information.
Default: No timestamp is included.

Usage Guidelines See “Specify the Files to Contain Traffic Sampling Output” on page 225.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

tftp

Syntax

```
tftp {
    description description-of-service;
    server address < [ routing-instance routing-instance-name ] >;
    interface interface-name {
        description description-of-interface;
        no-listen;
        server address < [ routing-instance routing-instance-name ] >;
    }
}
```

Hierarchy Level [edit forwarding-options helpers]

Description Enable TFTP request packet forwarding.

Options The statements are explained separately.

Usage Guidelines See “Configure DNS and TFTP Packet Forwarding” on page 220.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

traceoptions

traceoptions (for DNS and TFTP packet forwarding)

Syntax

```
traceoptions {
    file filename {
        files number;
        size bytes;
    }
    flag flag;
    level level;
}
```

Hierarchy Level [edit forwarding-options helpers]

Description Configure tracing operations.

Options flag—Tracing operation to perform. To specify more than one tracing operation, include multiple flag statements.

- address—Address management
- all—All forwarding tracing operations
- config—Configuration
- ifdb—Interface database
- io—I/O
- main—Main loop
- rtsock—Routing socket
- trace—Trace tracing
- ui—User interface
- util—Miscellaneous utility

level—Level at which traceoptions tracks the filter information:

- all—Match all levels.
- error—Match error conditions.
- info—Match informational messages.
- notice—Match conditions that should be handled specially.
- verbose—Match verbose messages.
- warning—Match warning messages.

The remaining statements are explained separately.

Usage Guidelines See “Configure DNS and TFTP Packet Forwarding” on page 220.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

traceoptions (for traffic sampling)

Syntax
```
traceoptions {
    file filename {
        files number;
        size bytes;
        (world-readable | no-world-readable);
    }
}
```

Hierarchy Level [edit forwarding-options sampling]

Description Configure traffic sampling tracing operations.

The statements are explained separately.

Usage Guidelines See “Trace Traffic Sampling Operations” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

version

Syntax version *format*;

Hierarchy Level [edit forwarding-options sampling output cflowd *host-name*]

Description Specify the version format of the aggregated flows exported to a cflowd server.

Options *format*—Format of the flows.
Values: 5 or 8
Default: 5

Usage Guidelines See “Configure Flow Aggregation (cflowd)” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

world-readable

Syntax (world-readable | no-world-readable);

Hierarchy Level [edit forwarding-options sampling output file],
[edit forwarding-options sampling traceoptions file]

Description Enable unrestricted file access.

Options no-world-readable—Restrict file access to owner. This is the default.

world-readable—Enable unrestricted file access.

Default: no-world-readable

Usage Guidelines See “Trace Traffic Sampling Operations” on page 227.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Part 5
Appendix

■ Glossary on page 261

Appendix A
Glossary

A

AAL ATM adaptation layer. A series of protocols enabling various types of traffic, including voice, data, image, and video, to run over an ATM network.

active route Route chosen from all routes in the routing table to reach a destination. Active routes are installed into the forwarding table.

add/drop multiplexer *See ADM*.

Address Resolution Protocol *See ARP*.

adjacency Portion of the local routing information that pertains to the reachability of a single neighbor over a single circuit or interface.

ADM Add/drop multiplexer. SONET functionality that allows lower-level signals to be dropped from a high-speed optical connection.

aggregation Combination of groups of routes that have common addresses into a single entry in the routing table.

AH Authentication Header. A component of the IPSec protocol used to verify that the contents of a packet have not been changed, and to validate the identity of the sender. *See also ESP*.

ANSI American National Standards Institute. The United States' representative to the ISO.

APQ Alternate Priority Queuing. Dequeuing method that has a special queue, similar to SPQ, which is visited only 50 percent of the time. The packets in the special queue still have a predictable latency, although the upper limit of the delay is higher than that with SPQ. Since the other configured queues share the remaining 50 percent of the service time, queue starvation is usually avoided. *See also SPQ*.

APS Automatic Protection Switching. Technology used by SONET ADMs to protect against circuit faults between the ADM and a router and to protect against failing routers.

area Routing subdomain that maintains detailed routing information about its own internal composition and that maintains routing information that allows it to reach other routing subdomains. In IS-IS, an area corresponds to a Level 1 subdomain.

In IS-IS and OSPF, a set of contiguous networks and hosts within an autonomous system that have been administratively grouped together.

area border router Router that belongs to more than one area. Used in OSPF.

ARP Address Resolution Protocol. Protocol for mapping IP addresses to MAC addresses.

AS Autonomous system. Set of routers under a single technical administration. Each AS normally uses a single interior gateway protocol (IGP) and metrics to propagate routing information within the set of routers. Also called *routing domain*.

AS boundary router In OSPF, routers that exchange routing information with routers in other ASs.

AS external link advertisements OSPF link-state advertisement sent by AS boundary routers to describe external routes that they know. These link-state advertisements are flooded throughout the AS (except for stub areas).

AS path In BGP, the route to a destination. The path consists of the AS numbers of all routers a packet must go through to reach a destination.

ASIC Application-specific integrated circuit. Specialized processors that perform specific functions on the router.

ATM Asynchronous Transfer Mode. A high-speed multiplexing and switching method utilizing fixed-length cells of 53 octets to support multiple types of traffic.

atomic Smallest possible operation. An atomic operation is performed either entirely or not at all. For example, if machine failure prevents a transaction from completing, the system is rolled back to the start of the transaction, with no changes taking place.

Authentication Header *See AH*.

Automatic Protection Switching *See APS*.

autonomous system *See AS*.

autonomous system boundary router In OSPF, routers that exchange routing information with routers in other ASs.

autonomous system external link advertisements OSPF link-state advertisement sent by autonomous system boundary routers to describe external routes that they know. These link-state advertisements are flooded throughout the autonomous system (except for stub areas).

autonomous system path In BGP, the route to a destination. The path consists of the autonomous system numbers of all the routers a packet must pass through to reach a destination.

B

backbone area In OSPF, an area that consists of all networks in area ID 0.0.0.0, their attached routers, and all area border routers.

backplane On an M40 router, component of the Packet Forwarding Engine that distributes power, provides signal connectivity, manages shared memory on FPCs, and passes outgoing data cells to FPCs.

bandwidth The range of transmission frequencies a network can use, expressed as the difference between the highest and lowest frequencies of a transmission channel. In computer networks, greater bandwidth indicates faster data-transfer rate capacity.

Bellcore Bell Communications Research. Research and development organization created after the divestiture of the Bell System. It is supported by the regional Bell holding companies (RBHCs), which own the regional Bell operating companies (RBOCs).

BERT — Bit error rate test. A test that can be run on a T3 interface to determine whether it is operating properly.

BGP — Border Gateway Protocol. Exterior gateway protocol used to exchange routing information among routers in different autonomous systems.

bit error rate test — *See BERT*.

BITS — Building Integrated Timing Source. Dedicated timing source that synchronizes all equipment in a particular building.

Border Gateway Protocol — *See BGP*.

broadcast — Operation of sending network traffic from one network node to all other network nodes.

bundle — Collection of software that makes up a JUNOS software release.

C

CB — Control Board. Part of the host subsystem that provides control and monitoring functions for router components.

CCC — Circuit cross-connect. A JUNOS software feature that allows you to configure transparent connections between two circuits, where a circuit can be a Frame Relay DLCI, an ATM VC, a PPP interface, a Cisco HDLC interface, or an MPLS label-switched path (LSP).

CE device — Customer edge device. Router or switch in the customer's network that is connected to a service provider's provider edge (PE) router and participates in a Layer 3 VPN.

CFM — Cubic feet per minute. Measure of air flow in volume per minute.

Challenge Handshake Authentication Protocol — *See CHAP*.

channel service unit — *See CSU/DSU*.

CHAP — A protocol that authenticates remote users. CHAP is a server-driven, three-step authentication mechanism that depends on a shared secret password that resides on both the server and the client.

CIDR — Classless interdomain routing. A method of specifying Internet addresses in which you explicitly specify the bits of the address to represent the network address instead of determining this information from the first octet of the address.

CIP — Connector Interface Panel. On an M160 router, the panel that contains connectors for the Routing Engines, BITS interfaces, and alarm relay contacts.

circuit cross-connect — *See CCC*.

class of service — *See CoS*.

CLEC — (Pronounced “see-lek”) Competitive Local Exchange Carrier. Company that competes with the already established local telecommunications business by providing its own network and switching.

CLEI — Common language equipment identifier. Inventory code used to identify and track telecommunications equipment.

CLI	Command-line interface. Interface provided for configuring and monitoring the routing protocol software.
client peer	In a BGP route reflection, a member of a cluster that is not the route reflector. *See also nonclient peer*.
CLNP	Connectionless Network Protocol. ISO-developed protocol for OSI connectionless network service. CLNP is the OSI equivalent of IP.
cluster	In BGP, a set of routers that have been grouped together. A cluster consists of one system that acts as a route reflector, along with any number of client peers. The client peers receive their route information only from the route reflector system. Routers in a cluster do not need to be fully meshed.
community	In BGP, a group of destinations that share a common property. Community information is included as one of the path attributes in BGP update messages.
confederation	In BGP, a group of systems that appears to external autonomous systems to be a single autonomous system.
constrained path	In traffic engineering, a path determined using RSVP signaling and constrained using CSPF. The ERO carried in the packets contains the constrained path information.
core	The central backbone of the network.
CoS	Class of service. The method of classifying traffic on a packet-by-packet basis using information in the ToS byte to provide different service levels to different traffic.
CPE	Customer premises equipment. Telephone or other service provider equipment located at a customer site.
craft interface	Mechanisms used by a Communication Workers of America craftsperson to operate, administer, and maintain equipment or provision data communications. On a Juniper Networks router, the craft interface allows you to view status and troubleshooting information and perform system control functions.
CSCP	Class Selector Codepoint.
CSNP	Complete sequence number PDU. Packet that contains a complete list of all the LSPs in the IS-IS database.
CSPF	Constrained Shortest Path First. An MPLS algorithm that has been modified to take into account specific restrictions when calculating the shortest path across the network.
CSU/DSU	Channel service unit/data service unit. Channel service unit connects a digital phone line to a multiplexer or other digital signal device. Data service unit connects a DTE to a digital phone line.
customer edge device	*See CE device*.

D

daemon Background process that performs operations on behalf of the system software and hardware. Daemons normally start when the system software is booted, and they run as long as the software is running. In the JUNOS software, daemons are also referred to as processes.

damping Method of reducing the number of update messages sent between BGP peers, thereby reducing the load on these peers without adversely affecting the route convergence time for stable routes.

data circuit-terminating equipment *See DCE*.

data-link connection identifier *See DLCI*.

data service unit *See CSU/DSU*.

Data Terminal Equipment *See DTE*.

dcd The JUNOS software interface process (daemon).

DCE Data circuit-terminating equipment. RS-232-C device, typically used for a modem or printer, or a network access and packet switching node.

default address Router address that is used as the source address on unnumbered interfaces.

denial of service *See DoS*.

dense wavelength-division multiplexing *See DWDM*.

designated router In OSPF, a router selected by other routers that is responsible for sending link-state advertisements that describe the network, which reduces the amount of network traffic and the size of the routers' topological databases.

destination prefix length Number of bits of the network address used for host portion of a CIDR IP address.

DHCP Dynamic Host Configuration Protocol. Allocates IP addresses dynamically so that they can be reused when they are no longer needed.

Diffie-Hellman A public key scheme, invented by Whitfield Diffie and Martin Hellman, used for sharing a secret key without communicating secret information, thus precluding the need for a secure channel. Once correspondents have computed the secret shared key, they can use it to encrypt communications.

Diffserv Differentiated Service (based on RFC 2474). Diffserv uses the ToS byte to identify different packet flows on a packet-by-packet basis. Diffserv adds a Class Selector Codepoint (CSCP) and a Differentiated Services Codepoint (DSCP).

Dijkstra algorithm *See SPF*.

DIMM Dual inline memory module. 168-pin memory module that supports 64-bit data transfer.

direct routes *See interface routes*.

DLCI — Data-link connection identifier. Identifier for a Frame Relay virtual connection (also called a logical interface).

DoS — Denial of service. System security breach in which network services become unavailable to users.

DRAM — Dynamic random-access memory. Storage source on the router that can be accessed quickly by a process.

drop profile — Drop probabilities for different levels of buffer fullness that are used by RED to determine from which queue to drop packets.

DSCP — Differentiated Services Codepoint.

DSU — Data service unit. A device used to connect a DTE to a digital phone line. Converts digital data from a router to voltages and encoding required by the phone line. *See also CSU/DSU*.

DTE — Data Terminal Equipment. RS-232-C interface that a computer uses to exchange information with a serial device.

DVMRP — Distance Vector Multicast Routing Protocol. Distributed multicast routing protocol that dynamically generates IP multicast delivery trees using a technique called reverse path multicasting (RPM) to forward multicast traffic to downstream interfaces.

DWDM — Dense wavelength-division multiplexing. Technology that enables data from different sources to be carried together on an optical fiber, with each signal carried on its own separate wavelength.

Dynamic Host Configuration Protocol — *See DHCP*.

E

EBGP — External BGP. BGP configuration in which sessions are established between routers in different ASs.

ECSA — Exchange Carriers Standards Association. A standards organization created after the divestiture of the Bell System to represent the interests of interexchange carriers.

edge router — In MPLS, a router located at the beginning or end of a label-switching tunnel. When at the beginning of a tunnel, an edge router applies labels to new packets entering the tunnel. When at the end of a tunnel, the edge router removes labels from packets exiting the tunnel. *See also MPLS*.

EGP — Exterior gateway protocol, such as BGP.

egress router — In MPLS, last router in a label-switched path (LSP). *See also ingress router*.

EIA — Electronic Industries Association. A United States trade group that represents manufacturers of electronics devices and sets standards and specifications.

EMI — Electromagnetic interference. Any electromagnetic disturbance that interrupts, obstructs, or otherwise degrades or limits the effective performance of electronics or electrical equipment.

encapsulating security payload — *See ESP*.

end system — In IS-IS, network entity that sends and receives packets.

ERO — Explicit Route Object. Extension to RSVP that allows an RSVP PATH message to traverse an explicit sequence of routers that is independent of conventional shortest-path IP routing.

ESP — Encapsulating security payload. A fundamental component of IPSec-compliant VPNs, ESP specifies an IP packet's encryption, data integrity checks, and sender authentication, which are added as a header to the IP packet. *See also AH.*

explicit path — *See signaled path*.

Explicit Route Object — *See ERO*.

export — To place routes from the routing table into a routing protocol.

external BGP — *See EBGP*.

external metric — A cost included in a route when OSPF exports route information from external autonomous systems. There are two types of external metrics: Type 1 and Type 2. Type 1 external metrics are equivalent to the link-state metric; that is, the cost of the route, used in the internal autonomous system. Type 2 external metrics are greater than the cost of any path internal to the autonomous system.

F

fast reroute — Mechanism for automatically rerouting traffic on an LSP if a node or link in an LSP fails, thus reducing the loss of packets traveling over the LSP.

FEAC — Far-end alarm and control. T3 signal used to send alarm or status information from the far-end terminal back to the near-end terminal and to initiate T3 loopbacks at the far-end terminal from the near-end terminal.

FEB — Forwarding Engine Board. In M5 and M10 routers, provides route lookup, filtering, and switching to the destination port.

firewall — A security gateway positioned between two different networks, usually between a trusted network and the Internet. A firewall ensures that all traffic that crosses it conforms to the organization's security policy. Firewalls track and control communications, deciding whether to pass, reject, discard, encrypt, or log them. Firewalls also can be used to secure sensitive portions of a local network.

FIFO — First in, first out.

flap damping — *See damping*.

flapping — *See route flapping*.

Flexible PIC Concentrator — *See FPC*.

Forwarding Engine Board — *See FEB*.

forwarding information base — *See forwarding table*.

forwarding table — JUNOS software forwarding information base (FIB). The JUNOS routing protocol process installs active routes from its routing tables into the Routing Engine forwarding table. The kernel copies this forwarding table into the Packet Forwarding Engine, which is responsible for determining which interface transmits the packets.

FPC — Flexible PIC Concentrator. An interface concentrator on which PICs are mounted. An FPC inserts into a slot in a Juniper Networks router. *See also PIC.*

FRU — Field-replaceable unit. Router component that customers can replace onsite.

G

group — A collection of related BGP peers.

H

hash — A one-way function that takes a message of any length and produces a fixed-length digest. In security, a message digest is used to validate that the contents of a message have not been altered in transit. The Secure Hash Algorithm (SHA-1) and Message Digest 5 (MD5) are commonly used hashes.

Hashed Message Authentication Code — *See HMAC*.

HDLC — High-level data link control. An International Telecommunication Union (ITU) standard for a bit-oriented data link layer protocol on which most other bit-oriented protocols are based.

HMAC — Hashed Message Authentication Code. A mechanism for message authentication that uses cryptographic hash functions. HMAC can be used with any iterative cryptographic hash function—for example, MD5 or SHA-1—in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

hold time — Maximum number of seconds allowed to elapse between the time a BGP system receives successive keepalive or update messages from a peer.

host module — On an M160 router, provides routing and system management functions of the router. Consists of the Routing Engine and Miscellaneous Control Subsystem (MCS).

host subsystem — Provides routing and system-management functions of the router. Consists of a Routing Engine and an adjacent Control Board (CB).

I

IANA — Internet Assigned Numbers Authority. Regulatory group that maintains all assigned and registered Internet numbers, such as IP and multicast addresses. *See also NIC*.

IBGP — Internal BGP. BGP configuration in which sessions are established between routers in the same ASs.

ICMP — Internet Control Message Protocol. Used in router discovery, ICMP allows router advertisements that enable a host to discover addresses of operating routers on the subnet.

IDE — Integrated Drive Electronics. Type of hard disk on the Routing Engine.

IEC — International Electrotechnical Commission. *See ISO*.

IEEE — Institute of Electronic and Electrical Engineers. International professional society for electrical engineers.

IETF — Internet Engineering Task Force. International community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

IGMP	Internet Group Membership Protocol. Used with multicast protocols to determine whether group members are present.
IGP	Interior gateway protocol, such as IS-IS, OSPF, and RIP.
IKE	Internet Key Exchange. The key management protocol used in IPSec, IKE combines the ISAKMP and Oakley protocols to create encryption keys and security associations.
import	To install routes from the routing protocols into a routing table.
ingress router	In MPLS, first router in a label-switched path (LSP). *See also egress router*.
inter-AS routing	Routing of packets among different ASs. *See also EBGP*.
intercluster reflection	In a BGP route reflection, the redistribution of routing information by a route reflector system to all nonclient peers (BGP peers not in the cluster). *See also route reflection*.
interface routes	Routes that are in the routing table because an interface has been configured with an IP address. Also called *direct routes*.
intermediate system	In IS-IS, network entity that sends and receives packets and that can also route packets.
internal BGP	*See IBGP*.
Internet Key Exchange	*See IKE*.
Internet Protocol Security	*See IPSec*.
Internet Security Association and Key Management Protocol	*See ISAKMP*.
intra-AS routing	The routing of packets within a single AS. *See also IBGP.*
IP	Internet Protocol. The protocol used for sending data from one point to another on the Internet.
IPSec	Internet Protocol Security. The industry standard for establishing VPNs, IPSec comprises a group of protocols and algorithms that provide authentication and encryption of data across IP-based networks.
ISAKMP	Internet Security Association and Key Management Protocol. A protocol that allows the receiver of a message to obtain a public key and use digital certificates to authenticate the sender's identity. ISAKMP is designed to be key exchange independent; that is, it supports many different key exchanges. *See also IKE and Oakley*.
IS-IS	Intermediate System-to-Intermediate System protocol. Link-state, interior gateway routing protocol for IP networks that also uses the shortest-path first (SPF) algorithm to determine routes.
ISO	International Organization for Standardization. Worldwide federation of standards bodies that promotes international standardization and publishes international agreements as International Standards.

ISP — Internet service provider. Company that provides access to the Internet and related services.

ITU — International Telecommunications Union (formerly known as the CCITT). Group supported by the United Nations that makes recommendations and coordinates the development of telecommunications standards for the entire world.

J

jitter — Small random variation introduced into the value of a timer to prevent multiple timer expirations from becoming synchronized.

K

kernel forwarding table — *See forwarding table*.

L

label — In MPLS, 20-bit unsigned integer in the range 0 through 1048575, used to identify a packet traveling along an LSP.

label-switched path (LSP) — Sequence of routers that cooperatively perform MPLS operations for a packet stream. The first router in an LSP is called the *ingress router*, and the last router in the path is called the *egress router*. An LSP is a point-to-point, half-duplex connection from the ingress router to the egress router. (The ingress and egress routers cannot be the same router.)

label switching — *See MPLS*.

label-switching router — *See LSR*.

link — Communication path between two neighbors. A link is *up* when communication is possible between the two end points.

link-state PDU (LSP) — Packets that contain information about the state of adjacencies to neighboring systems.

local preference — Optional BGP path attribute carried in internal BGP update packets that indicates the degree of preference for an external route.

loose — In the context of traffic engineering, a path that can use any route or any number of other intermediate (transit) points to reach the next address in the path. (Definition from RFC 791, modified to fit LSPs.)

LSP — *See label-switched path (LSP)*) *or link-state PDU (LSP)*.

LSR — Label-switching router. A router on which MPLS and RSVP are enabled and is thus capable of processing label-switched packets.

M

martian address — Network address about which all information is ignored.

mask — *See subnet mask*.

MBGP — Multiprotocol BGP. An extension to BGP that allows you to connect multicast topologies within and between BGP ASs.

MBone — Internet multicast backbone. An interconnected set of subnetworks and routers that support the delivery of IP multicast traffic. The MBone is a virtual network that is layered on top of sections of the physical Internet.

MCS Miscellaneous Control Subsystem. On an M160 router, provides control and monitoring functions for router components and SONET clocking for the router.

MD5 Message Digest 5. A one-way hashing algorithm that produces a 128-bit hash. It is used in AH and ESP. *See also SHA-1*.

MDRR Modified Deficit Round Robin. A method for selecting queues to be serviced.

MED Multiple exit discriminator. Optional BGP path attribute consisting of a metric value that is used to determine the exit point to a destination when all other factors in determining the exit point are equal.

mesh Network topology in which devices are organized in a manageable, segmented manner with many, often redundant, interconnections between network nodes.

Message Digest 5 *See MD5*.

MIB Management Information Base. Definition of an object that can be managed by SNMP.

midplane Forms the rear of the PIC cage on M5 and M10 routers and the FPC card cage on M20 and M160 routers. Provides data transfer, power distribution, and signal connectivity.

Miscellaneous Control Subsystem *See MCS*.

MPLS Multiprotocol Label Switching. Mechanism for engineering network traffic patterns that functions by assigning to network packets short labels that describe how to forward them through the network. Also called *label switching. See also traffic engineering.*

MTBF Mean time between failure. Measure of hardware component reliability.

MTU Maximum transfer unit. Limit on segment size for a network.

multicast Operation of sending network traffic from one network node to multiple network nodes.

multicast distribution tree The data path between the sender (host) and the multicast group member (receiver or listener).

multiprotocol BGP *See MBGP*.

Multiprotocol Label Switching *See MPLS*.

N

neighbor Adjacent system reachable by traversing a single subnetwork. An immediately adjacent router. Also called a *peer.*

NET Network entity title. Network address defined by the ISO network architecture and used in CLNS-based networks.

network layer reachability information *See NLRI*.

network link advertisement An OSPF link-state advertisement flooded throughout a single area by designated routers to describe all routers attached to the network.

Network Time Protocol — *See NTP*.

NIC — Network Information Center. Internet authority responsible for assigning Internet-related numbers, such as IP addresses and autonomous system numbers. *See also IANA*.

NLRI — Network layer reachability information. Information that is carried in BGP packets and is used by MBGP.

nonclient peer — In a BGP route reflection, a BGP peer that is not a member of a cluster. *See also client peer*.

not-so-stubby area — *See NSSA*.

NSAP — Network service access point. Connection to a network that is identified by a network address.

n-selector — Last byte of an nonclient peer address.

NSSA — Not-so-stubby area. In OSPF, a type of stub area in which external routes can be flooded.

NTP — Network Time Protocol. Protocol used to synchronize computer clock times on a network.

O

Oakley — A key determination protocol based on the Diffie-Hellman algorithm that provides added security, including authentication. Oakley was the key-exchange algorithm mandated for use with the initial version of ISAKMP, although various algorithms can be used. Oakley describes a series of key exchanges called “modes” and details the services provided by each; for example, Perfect Forward Secrecy for keys, identity protection, and authentication. *See also ISAKMP*.

OC — Optical Carrier. In SONET, Optical Carrier levels indicate the transmission rate of digital signals on optical fiber.

OSI — Open System Interconnection. Standard reference model for how messages are transmitted between two points on a network.

OSPF — Open Shortest Path First. A link-state IGP that makes routing decisions based on the shortest-path-first (SPF) algorithm (also referred to as the *Dijkstra algorithm*).

P

package — A collection of files that make up a JUNOS software component.

Packet Forwarding Engine — The architectural portion of the router that processes packets by forwarding them between input and output interfaces.

path attribute — Information about a BGP route, such as the route origin, AS path, and next-hop router.

PCI — Peripheral Component Interconnect. Standard, high-speed bus for connecting computer peripherals. Used on the Routing Engine.

PCMCIA — Personal Computer Memory Card International Association. Industry group that promotes standards for credit card-size memory or I/O devices.

PDU — Protocol data unit. IS-IS packets.

PE router — Provider edge router. A router in the service provider's network that is connected to a customer edge (CE) device and that participates in a Virtual Private Network (VPN).

PEC — Policing Equivalence Classes. In traffic policing, a set of packets that is treated the same by the packet classifier.

peer — An immediately adjacent router with which a protocol relationship has been established. Also called a *neighbor*.

Perfect Forward Secrecy — *See PFS*.

PFE — *See Packet Forwarding Engine*.

PFS — A condition derived from an encryption system that changes encryption keys often and ensures that no two sets of keys have any relation to each other. The advantage of PFS is that if one set of keys is compromised, only communications using those keys are at risk. An example of a system that uses PFS is Diffie-Hellman.

Physical Interface Card — *See PIC*.

PIC — Physical Interface Card. A network interface–specific card that can be installed on an FPC in the router.

PIM — Protocol Independent Multicast. A protocol-independent multicast routing protocol. PIM Sparse Mode routes to multicast groups that might span wide-area and interdomain internets. PIM Dense Mode is a flood-and-prune protocol.

PLP — Packet Loss Priority.

PLP bit — Packet Loss Priority bit. Used to identify packets that have experienced congestion or are from a transmission that exceeded a service provider's customer service license agreement. This bit can be used as part of a router's congestion control mechanism and can be set by the interface or by a filter.

policing — Applying rate limits on bandwidth and burst size for traffic on a particular interface.

pop — Removal of the last label, by a router, from a packet as it exits an MPLS domain.

PPP — Point-to-Point Protocol. Link-layer protocol that provides multiprotocol encapsulation. It is used for link-layer and network-layer configuration.

precedence bits — The first three bits in the ToS byte. On a Juniper Networks router, these bits are used to sort or classify individual packets as they arrive at an interface. The classification determines the queue to which the packet is directed upon transmission.

preference — Desirability of a route to become the active route. A route with a lower preference value is more likely to become the active route. The preference is an arbitrary value in the range 0 through 255 that the routing protocol process uses to rank routes received from different protocols, interfaces, or remote systems.

preferred address — On an interface, the default local address used for packets sourced by the local router to destinations on the subnet.

primary address — On an interface, the address used by default as the local address for broadcast and multicast packets sourced locally and sent out the interface.

primary interface — Router interface that packets go out when no interface name is specified and when the destination address does not imply a particular outgoing interface.

Protocol-Independent Multicast	*See PIM*.
provider edge router	*See PE router*.
provider router	Router in the service provider's network that does not attach to a customer edge (CE) device.
PSNP	Partial sequence number PDU. Packet that contains only a partial list of the LSPs in the IS-IS link-state database.
push	Addition of a label or stack of labels, by a router, to a packet as it enters an MPLS domain.

Q

QoS	Quality of service. Performance, such as transmission rates and error rates, of a communications channel or system.
quality of service	*See QoS*.

R

RADIUS	Remote Authentication Dial-In User Service. Authentication method for validating users who attempt to access the router using Telnet.
Random Early Detection	*See RED*.
rate limiting	*See policing*.
RBOC	(Pronounced "are-bock") Regional Bell operating company. Regional telephone companies formed as a result of the divestiture of the Bell System.
RDRAM	RAMBUS dynamic random access memory.
RED	Random Early Detection. Gradual drop profile for a given class that is used for congestion avoidance. RED tries to anticipate incipient congestion and reacts by dropping a small percentage of packets from the head of the queue to ensure that a queue never actually becomes congested.
Rendezvous Point	*See RP*.
Resource Reservation Protocol	*See RSVP*.
RFC	Request for Comments. Internet standard specifications published by the Internet Engineering Task Force.
RFI	Radio frequency interference. Interference from high-frequency electromagnetic waves emanating from electronic devices.
RIP	Routing Information Protocol. Distance-vector interior gateway protocol that makes routing decisions based on hop count.
route flapping	Situation in which BGP systems send an excessive number of update messages to advertise network reachability information.
route identifier	IP address of the router from which a BGP, IGP, or OSPF packet originated.

route reflection — In BGP, configuring a group of routers into a cluster and having one system act as a route reflector, redistributing routes from outside the cluster to all routers in the cluster. Routers in a cluster do not need to be fully meshed.

router link advertisement — OSPF link-state advertisement flooded throughout a single area by all routers to describe the state and cost of the router's links to the area.

routing domain — *See AS*.

Routing Engine — Architectural portion of the router that handles all routing protocol processes, as well as other software processes that control the router's interfaces, some of the chassis components, system management, and user access to the router.

routing instance — A collection of routing tables, interfaces, and routing protocol parameters. The set of interfaces belongs to the routing tables and the routing protocol parameters control the information in the routing tables.

routing table — Common database of routes learned from one or more routing protocols. All routes are maintained by the JUNOS routing protocol process.

RP — For PIM-SM, a core router acting as the root of the distribution tree in a shared tree.

rpd — JUNOS software routing protocol process (daemon). User-level background process responsible for starting, managing, and stopping the routing protocols on a Juniper Networks router.

RPM — Reverse path multicasting. Routing algorithm used by DVMRP to forward multicast traffic.

RSVP — Resource Reservation Protocol. Resource reservation setup protocol designed to interact with integrated services on the Internet.

S

SA — Security association. An IPSec term that describes an agreement between two parties about what rules to use for authentication and encryption algorithms, key exchange mechanisms, and secure communications.

SAP — Session Announcement Protocol. Used with multicast protocols to handle session conference announcements.

SAR — Segmentation and reassembly. Buffering used with ATM.

SCB — System Control Board. On an M40 router, the part of the Packet Forwarding Engine that performs route lookups, monitors system components, and controls FPC resets.

SCG — SONET Clock Generator. Provides Stratum 3 clock signal for the SONET/SDH interfaces on the router. Also provides external clock inputs.

SDH — Synchronous Digital Hierarchy. CCITT variation of SONET standard.

SDP — Session Description Protocol. Used with multicast protocols to handle session conference announcements.

SDRAM — Synchronous dynamic random access memory.

Secure Hash Algorithm — *See SHA-1*.

secure shell — *See SSH*.

security association	*See SA*.
Security Parameter Index	*See SPI*.
SFM	Switching and Forwarding Module. On an M160 router, a component of the Packet Forwarding Engine that provides route lookup, filtering, and switching to FPCs.
SHA-1	Secure Hash Algorithm. A widely used hash function for use with Digital Signal Standard (DSS). SHA-1 is more secure than MD5.
shortest-path-first algorithm	*See SPF*.
signaled path	In traffic engineering, an explicit path; that is, a path determined using RSVP signaling. The ERO carried in the packets contains the explicit path information.
SIB	Switch Interface Board. Provides the switching function to the destination Packet Forwarding Engine.
simplex interface	An interface that assumes that packets it receives from itself are the result of a software loopback process. The interface does not consider these packets when determining whether the interface is functional.
SNMP	Simple Network Management Protocol. Protocol governing network management and the monitoring of network devices and their functions.
SONET	Synchronous Optical Network. High-speed (up to 2.5 Gbps) synchronous network specification developed by Bellcore and designed to run on optical fiber. STS-1 is the basic building block of SONET. Approved as an international standard in 1988. *See also SDH.*
SPF	Shortest-path first, an algorithm used by IS-IS and OSPF to make routing decisions based on the state of network links. Also called the *Dijkstra algorithm*.
SPI	Security Parameter Index. A portion of the IPSec Authentication Header that communicates which security protocols, such as authentication and encryption, are used for each packet in a VPN connection.
SPQ	Strict Priority Queuing. Dequeuing method that provides a special queue that is serviced until it is empty. The traffic sent to this queue tends to maintain a lower latency and more consistent latency numbers than traffic sent to other queues. *See also APQ.*
SSB	System and Switch Board. On an M20 router, Packet Forwarding Engine component that performs route lookups and component monitoring and monitors FPC operation.
SSH	Secure shell. Software that provides a secured method of logging in to a remote network system.
SSRAM	Synchronous Static Random Access Memory.
static LSP	*See static path*.
static path	In the context of traffic engineering, a static route that requires hop-by-hop manual configuration. No signaling is used to create or maintain the path. Also called a *static LSP.*
STM	Synchronous Transport Module. CCITT specification for SONET at 155.52 Mbps.

strict In the context of traffic engineering, a route that must go directly to the next address in the path. (Definition from RFC 791, modified to fit LSPs.)

STS Synchronous Transport Signal. Synchronous Transport Signal level 1. Basic building block signal of SONET, operating at 51.84 Mbps. Faster SONET rates are defined as STS-*n*, where *n* is a multiple of 51.84 Mbps. *See also SONET*.

stub area In OSPF, an area through which, or into which, AS external advertisements are not flooded.

subnet mask Number of bits of the network address used for host portion of a Class A, Class B, or Class C IP address.

summary link advertisement OSPF link-statement advertisement flooded throughout the advertisement's associated areas by area border routers to describe the routes that they know about in other areas.

sysid System identifier. Portion of the ISO nonclient peer. The sysid can be any 6 bytes that are unique throughout a domain.

System and Switch Board *See SSB*.

T

TACACS + Terminal Access Controller Access Control System Plus. Authentication method for validating users who attempt to access the router using Telnet.

TCP Transmission Control Protocol. Works in conjunction with Internet Protocol (IP) to send data over the Internet. Divides a message into packets and tracks the packets from point of origin to destination.

ToS Type of service. The method of handling traffic using information extracted from the fields in the ToS byte to differentiate packet flows.

traffic engineering Process of selecting the paths chosen by data traffic in order to balance the traffic load on the various links, routers, and switches in the network. (Definition from http://www.ietf.org/internet-drafts/draft-ietf-mpls-framework-04.txt.) *See also MPLS*.

transit area In OSPF, an area used to pass traffic from one adjacent area to the backbone or to another area if the backbone is more than two hops away from an area.

transit router In MPLS, any intermediate router in the LSP between the ingress router and the egress router.

transport mode An IPSec mode of operation in which the data payload is encrypted, but the original IP header is left untouched. The IP addresses of the source or destination can be modified if the packet is intercepted. Because of its construction, transport mode can be used only when the communication endpoint and cryptographic endpoint are the same. VPN gateways that provide encryption and decryption services for protected hosts cannot use transport mode for protected VPN communications. *See also tunnel mode*.

Triple-DES A 168-bit encryption algorithm that encrypts data blocks with three different keys in succession, thus achieving a higher level of encryption. Triple-DES is one of the strongest encryption algorithms available for use in VPNs.

tunnel Private, secure path through an otherwise public network.

tunnel mode An IPSec mode of operation in which the entire IP packet, including the header, is encrypted and authenticated and a new VPN header is added, protecting the entire original packet. This mode can be used by both VPN clients and VPN gateways, and protects communications that come from or go to non-IPSec systems. *See also transport mode*.

Tunnel PIC A physical interface card that allows the router to perform the encapsulation and decapsulation of IP datagrams. The Tunnel PIC supports IP-IP, GRE, and PIM register encapsulation and decapsulation. When the Tunnel PIC is installed, the router can be a PIM rendezvous point (RP) or a PIM first-hop router for a source that is directly connected to the router.

type of service *See ToS*.

U

unicast Operation of sending network traffic from one network node to another individual network node.

UPS Uninterruptible power supply. Device that sits between a power supply and a router (or other piece of equipment) the prevents undesired power-source events, such as outages and surges, from affecting or damaging the device.

V

vapor corrosion inhibitor *See VCI*.

VCI Vapor corrosion inhibitor. Small cylinder packed with the router that prevents corrosion of the chassis and components during shipment.

VCI Virtual circuit identifier. 16-bit field in the header of an ATM cell that indicates the particular virtual circuit the cell takes through a virtual path. Also called a *logical interface. See also VPI*.

virtual circuit identifier *See VCI*.

virtual link In OSPF, a link created between two routers that are part of the backbone but are not physically contiguous.

virtual path identifier *See VPI*.

virtual private network *See VPN*.

Virtual Router Redundancy Protocol *See VRRP*.

VPI virtual path identifier. 8-bit field in the header of an ATM cell that indicates the virtual path the cell takes. *See also VCI*.

VPN virtual private network. A private data network that makes use of a public TCP/IP network, typically the Internet, while maintaining privacy with a tunneling protocol, encryption, and security procedures.

VRRP Virtual Router Redundancy Protocol. On Fast Ethernet and Gigabit Ethernet interfaces, allows you to configure virtual default routers.

W

wavelength-division multiplexing — *See WDM*.

WDM — Wavelength-division multiplexing. Technique for transmitting a mix of voice, data, and video over various wavelengths (colors) of light.

WFQ — Weighted Fair Queuing.

weighted round-robin — *See WRR*.

WRR — Weighted round-robin. Scheme used to decide the queue from which the next packet should be transmitted.

Part 6
Indexes

Index

Index

Bold numbers point to command summary pages.

Index of Statements and Commands